SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Tomislav Horvat

Zagreb, 2013.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor: Student:

Prof. dr. sc. Mladen Crneković, dipl. ing. Tomislav Horvat

Zagreb, 2013.

Izjavljujem da sam ovaj rad izradio samostalno koristeći stečena znanja tijekom studija i navedenu literaturu.

Zahvaljujem se profesorima Mladenu Crnekoviću i Josipu Kasaću na korisnim savjetima koje su mi dali tijekom izrade ovog rada.

Posebno se zahvaljujem Petri Gavranović bez čije podrške i razumijevanja ovaj rad ne bi bio ostvaren.

Tomislav Horvat

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

SAŽETAK

Ovim radom zahvaćena je problematika prepoznavanja objekata pomoću vizijskih sustava u usporedbi s ljudskim prepoznavanjem objekata. Izrađen je vizijski sustav koji koristi web-kameru montiranu na robotu za prepoznavanje vrste i boje objekata te njihovo sortiranje. Prikazana i objašnjena je primjena RGB filtera, tri procedure sivljenja, Robert Cross operatora i filtracije smetnji pomoću veličine objekta na primljenoj slici. Također, izložene su dvije procedure za traženje krajnjih točki objekta i njegova prepoznavanja te dvije procedure za prepoznavanje boje objekta na slici. Na kraju rada dana su detaljna uputstva za upotrebu programa.

1. UVOD

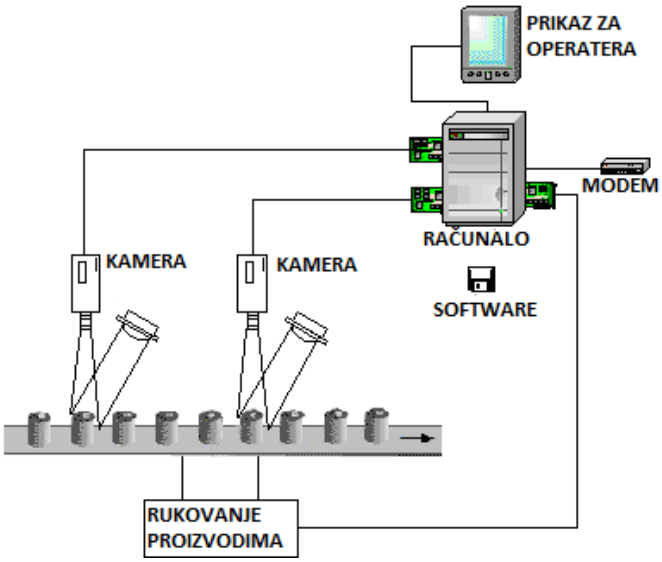
Ljudski vidni sustav sastoji se od oka, očnog živca i mozga. Oko služi za primanje slike, a očni živac za prijenos slike do mozga u kojem se vrši obrada slike. Računalni vid je pokušaj oponašanja ljudskog vidnog sustava s ciljem davanja vida računalu.

Računalni vid obuhvaća procedure i metode za primanje, procesiranje, analiziranje i prevođenje slika iz stvarnog svijeta u računalu prihvatljive matematičke ili simboličke informacije iz kojih računalo može, na temelju programiranog algoritma, odlučiti koju će programiranu radnju izvršiti. Računalni vid je multidisciplinarno područje koje obuhvaća robotsku viziju, kompjutersku inteligenciju, kognitivnu viziju, statistiku, geometriju, optimizaciju, optiku i mnoga druga područja. Vizijski sustavi su cjeline koje se sastoje od prihvatnog člana (npr. kamere), mjesta na kojem će se vršiti obrada računalnog vida (npr. stolno računalo) te nekog izvršnog člana (npr. robot).

Područja primjene vizijskih sustava su:

- prilikom automatizacije tvornica – npr. navođenje robota, provjera ispravnosti proizvoda;
- pri praćenju ljudi i vozila – npr. automatsko snimanje tablica prilikom prekoračenja brzine;
- korištenje kamere kao senzora – npr. paljenje svjetla kod detekcije prisutnosti čovjeka;
- pri prepoznavanju obrasca – npr. očitavanje bar kodova;
- pri izradi 3D CAD modela na temelju 2D slika – npr. Tritop i Atos sustavi; [1]
- pri mjerenju dimenzija objekata – npr. Tritop i Atos sustavi; [1]
- u vojnoj industriji – npr. praćenje i gađanje zadanog objekta;
- u industriji računalnih igara – npr. Kinect. [2]

Glavni razlog uvođenja vizijskih sustava u industriju je taj da ljudi ne mogu osigurati potrebnu ponovljivost pri obavljanju monotonih poslova čime narušavaju krajnju kvalitetu proizvoda. Pravilno projektiran robotski sustav koji informacije prima s vizijskog sustava može obavljati monotone poslove bez ikakve pogreške. Drugi razlog uvođenja je humanizacija rada. Humanizacija rada zahtijeva da se ljudska ili životinjska radna snaga zamijeni umjetnom – strojevima, kod svih poslova koji mogu dovesti dovesti u direktnu vezu s narušavanjem zdravlja. [3]



Slika 1. Industrijski vizijski sustav [4]

1.1. Razlika između ljudskog i računalnog vida

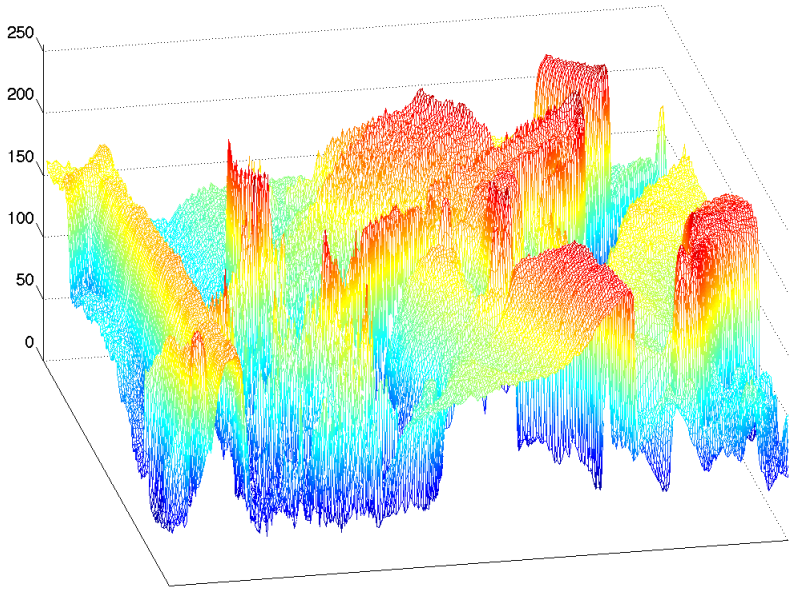
Ljudi u slikama prepoznaju određene otprije naučene strukture (npr. predmete, ljude, životinje) relativno jednostavno, nasuprot računalu koje vidi sliku kao matricu popunjenu RGB vrijednostima (*eng. Red Green Blue – Crveno Zeleno Plavo*). Glavni cilj računalnog vida jest dati računalima vid što sličniji ljudskom. Slika 2.a) prikazuje jedno od mogućih ljudskih interpretacija slike na kojoj su prikazane strukture pozadine, šešira, žene, zrcala i odraza u zrcalu, nasuprot računalnoj interpretaciji 2.b) iz koje je čak i čovjeku nemoguće zaključiti o kakvoj se slici radi. [5]

Glavni problem računalnog vida jest na koji način pristupiti informacijama slike kako bismo iz njih dobili korisne informacije. Prvi pristup ovom problemu bio je traženje informacija u susjednim pikselima. Taj pristup daje izvrsne informacija ako je potrebno pronaći konturu objekta koji robot treba prihvatiti, provjeriti postoji li zračnost između dva ugradbena dijela ili ako radimo kontrolu kvalitete pa tražimo nesavršenosti na proizvodima, ali ne daje nikakve informacije ako tražimo da nam locira što je pozadina a što čovjek. Pristup koji nam ovdje treba je traženje konture objekata i njihovo grupiranje na temelju njihovih atributa. Ovakav algoritam može raspoznati radi li se o pozadini, čovjeku ili zrcalu. Problem ovakvog pristupa je razina njegove detaljnosti; da, on može prepoznati da se radi o čovjeku, ali ne može identificirati dijelove kao što su oko, kosa, usta, lice ili ruka. Za takvu identifikaciju objekata sa slike potrebno je koristiti najvišu razinu prepoznavanja koja koristi

razne statističke metode prepoznavanja uzorka i klasifikacije. Tim pristupom možemo dobiti informacije o sceni slike sa svim objektima u slici. Ovakvi pristupi ne koriste se pri *online* prepoznavanju objekta jer su suviše spori, tako da će se u ovom radu koristiti prvi pristup.

a) [6]

b)

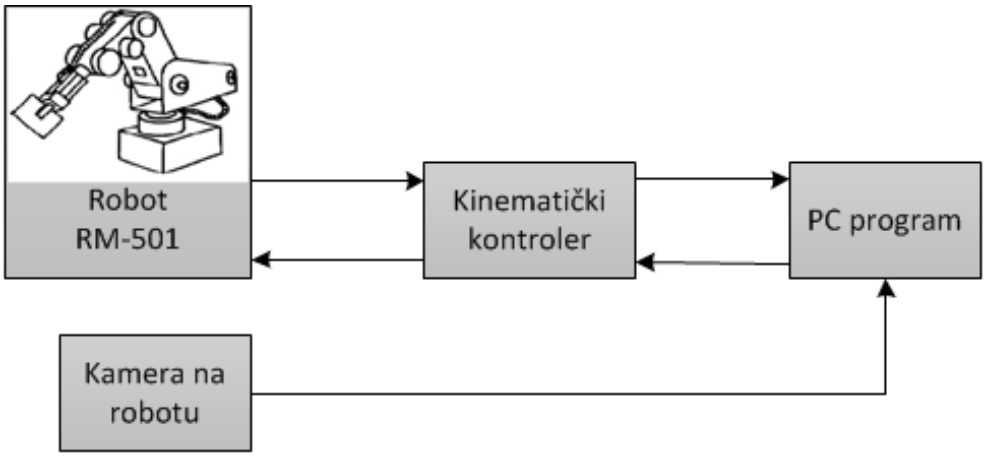


c)

Slika 2. Ljudsko nasuprot računalnom raspoznavanju slika

2. VIZIJSKI SUSTAV MITSUBISHI ROBOTA RM501

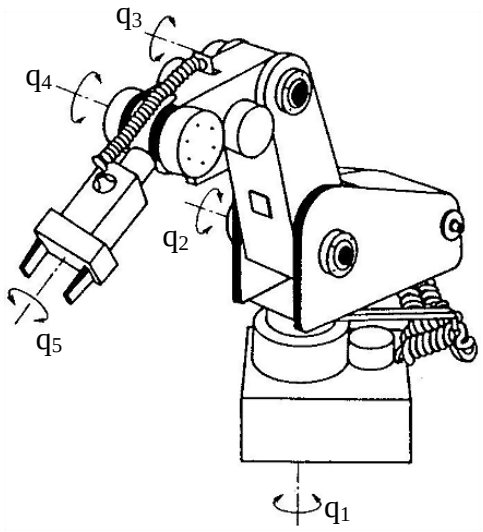
Vizijski sustav sastoji se od Mitsubishi robota RM501, kinematičkog kontrolera, kamere na robotu i računala.



Slika 3. Struktura vizijskog sustava

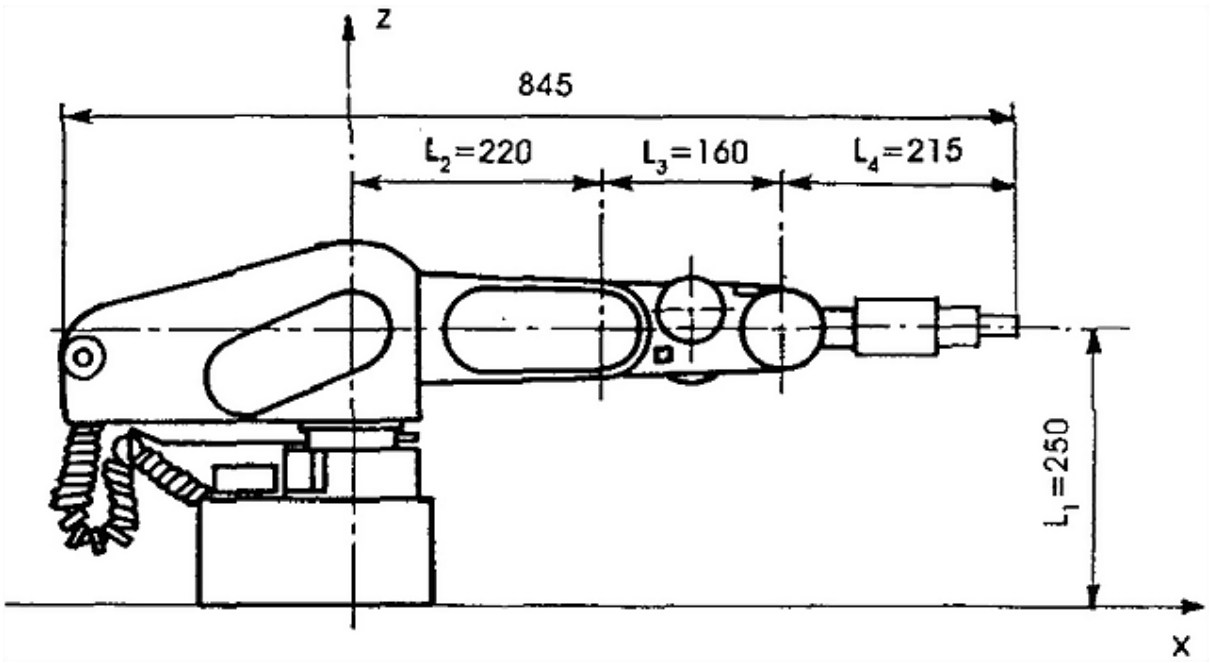
2.1. Podaci o robotu Mitsubishi RM501

Mitsubishi RM501 ima pet stupnjeva slobode. Točnost ponavljanja mu iznosi 0,15 mm. Pogon mu se sastoji od istosmjernih motora s napajanjem od 24 V te ukupne snage od 15 W. Prijenosni pogon između zglobova i motora ostvaren je pomoću zupčanika i remena. Maksimalna brzina prihvatnice iznosi 400 mm/s. Upravljanje je moguće vršiti pomoću *Point-To-Point* metode ili po glavnim *XYZ* koordinatnim osima robota te je kod tih načina rada moguće pohraniti makismalno 629 točaka u prostoru. Moguće je programirati kretanje robota po pohranjenim točkama gdje maksimalan broj naredbi za kretanja robota iznosi 2048. Razlučivost pozicija zglobova q_1, q_2, q_3 iznosi 0,025° dok razlučivost zglobova q_4, q_5 iznosi 0,075°. Robot se sastoji od energetskog i informacijskog dijela za koje robot ima zasebne priključne kablove. Robot ima sljedeće priključke: printerski paralelni ulaz, serijski RS-232C, privjesak za učenje i 8-bitni digitalni ulaz i izlaz. Nosivost robota bez prihvatnice iznosi 1,2 kg, s trenutno montiranom prihvatnicom 0,5 kg, no prihvatnice je moguće mijenjati. Masa trenutno montirane prihvatnice iznosi 0,7 kg, njena maksimalna otvorenost iznosi 60 mm a maksimalna sila pritiska iznosi 44 N. [7]



q_1 = struk	300°
q_2 = rame	130°
q_3 = lakat	90°
q_4 = nagib	± 90°
q5 = valjanje	± 180°

Slika 4. Maksimalno kretanje robota Mitsubishi RM501 u zglobovima [7]



Slika 5. Dimenzije robota Mitsubishi RM501 [7]

2.2. Podaci o kinematičkom kontroleru

Svrha kinematičkog kontrolera jest unaprijediti i olakšati način komunikacije s robotom RM501. Program unutar kinematičkog kontrolera obavlja izračun kinematičkog problema za zadane koordinate od strane korisnika na računalu. Nakon izračunatog

kinematičkog problema, program preko paralelne veze šalje robotu podatke za novu poziciju te se on potom pomakne u nju. Program također ima mogućnost vraćanja vanjskih i unutarnjih koordinata korisniku, kao i slanje drugih osnovnih naredbi robotu.

Kinematički kontroler temelji se na radu njegova glavnog mikrokontrolera oznake AT89C51ID2 tvrtke Atmel. Mikrokontroleri te serije tvrtke Atmel temelje se na seriji mikroprocesora Intel 8051. Mikrokontroler AT89C51ID2 je 8-bitni kontroler sa 64 kB flash-memorije, 2 kB EEPROM memorije te 32 izlazno/ulazne fizičke nožice. Mikrokontroler ima tri 16-bitna *timera*, te jedan *watchdog timer*.

Na kinematičkom kontroleru potrebno je spojiti se preko serijske veze. Zbog toga što većina današnjih stolnih i prijenosnih računala ne posjeduje serijsko sučelje, potrebno je koristiti pretvornik USB-veze u serijsku. USB-kraj pretvornika se spoji na računalo, dok se drugi kraj spoji na RS232 sučelje kinematičkog kontrolera. Kinematički kontroler se dalje spaja na upravljačku jedinicu robota preko paralelnog sučelja (serijsko sučelje koje se nalazi na upravljačkoj jedinici nije u funkciji). [8]

Slika 6. Kinematički kontroler robota Mitsubishi RM501

U kinematičkom kontroleru postoji tridesetak isprogramiranih naredbi kojima je moguće upravljati robotom te će kinematički kontroler kao odgovor na te naredbe poslati jednu od devet vrsta pogrešaka (*eng. error*), od kojih E0 znači da je kinematički kontroler zadanu naredbu izvršio uspješno.

2.3. Podaci o montiranoj kameri

Prihvat slike vrši se pomoću web-kamere koja je modificirana tako da ju je moguće pričvrstiti na držač kamere koji se nalazi na robotu. Korišteni model kamere je Canyon CNR-WCAM513.

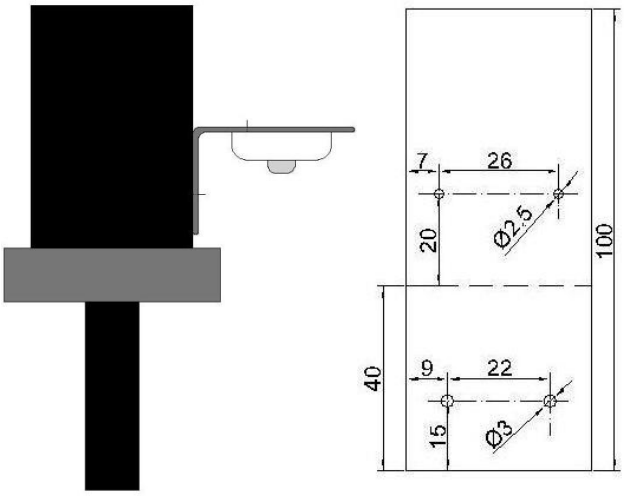
Karakteristike kamere su:

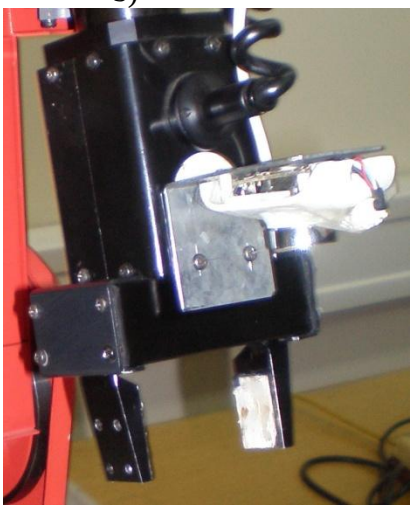
- rezulucija senzora – 1,3 Mpiksela;
- granica fokusa – od 5 cm nadalje;
- najveći frame-rate – 30 *frameova* u sekundi pri rezoluciji od 640 x 480 piksela;
- ručno podešavanje fokusa. [9]

Slika 7. prikazuje nosač kamere pomoću kojeg je web-kamera montirana na robot. On je izrađen iz lima debljine 2 mm savijanjem.

a) [9]

b)





c)

Slika 7. Web-kamera Canyon CNR-WCAM513

2.4. Podaci o PC-programu

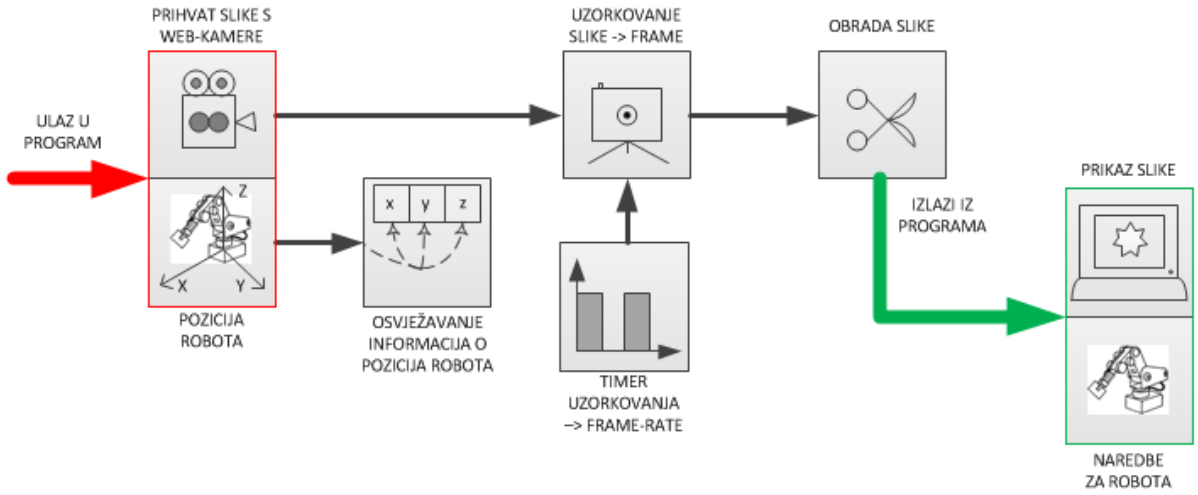
Računalo na kojem je programiran i testiran sustav prepoznavanja ima sljedeće karakteristike:

- procesor: 2 GHz;
- radna memorija : 2 Gb;
- integrirana grafička kartica: 256 Mb;
- operativni sustav: Windows XP – 32-bitni.

Računalni program koji se koristi za primanje i obradu slike te slanje naredbi kinematičkom kontroleru je Embarcadero Delphi 2010 – tridesetodnevna evaluacijska verzija. Programski paket korišten za prihvat slike s web-kamere unutar programskog sučelja Delphija je Video-lab tvrtke Mitov. Paket dolazi s mnogim gotovim funkcijama od kojih se za prihvat slike koristi DSCapture koji vrši prihvat slike pomoću DirectShow Windows *drivera* koji su sastavni dio novijih DirectX *drivera*. Programski paket koji se koristi za povezivanje Delphi programa sa serijskim sučeljem kinematičkog kontrolera je CPort Library. Za primanje i slanje podataka unutar programa koristi se ComPort komponenta. Pomoću nje moguće je namjestiti na kojem će Windows *portu* program primati i slati informacije (*eng. port*), kolika će biti brzina prijenosa podataka (*eng. baud rate*), koliko će bitova zauzimati jedan podatak (*eng. data bits*), kakav će biti zaustavni bit (*eng. stop bit*) i paritet podataka (*eng. parity*). [10] [11]

3. Sustav za sortiranje objekata robotom RM501

Sustav sortiranja objekata sastoji se od nekoliko podsustava koji rade zajedno. Kao ulaz u sustav sortiranja dobivamo sliku s web-kamere te poziciju robota. Kao izlaze iz sustava dobivamo modificiranu sliku koju prikazujemo korisniku na zaslonu računala te naredbe robotu koje program šalje kinematičkom kontroleru.



Slika 8. Struktura programa za sortiranje objekata robotom RM501

Strukturu prvog filtera koristimo za kalibraciju kamere vizijskog sustava, manipuliranje slikom pomoću različitih filtera ili Robert Cross operatorom te za crtanje pomoćnih linija. Indirektno, strukturu prvog filtera koristimo za prepoznavanje boja predmeta i za transformaciju koordinatnog sustava kamere. Strukturu drugog filtera koristimo prilikom izvođenja automatskog prepoznavanja i sortiranja pronađenih objekata na slici. Strukturu trećeg filtera koristimo za snimanje primljenih slika koje se tada mogu koristiti za *offline* analizu.

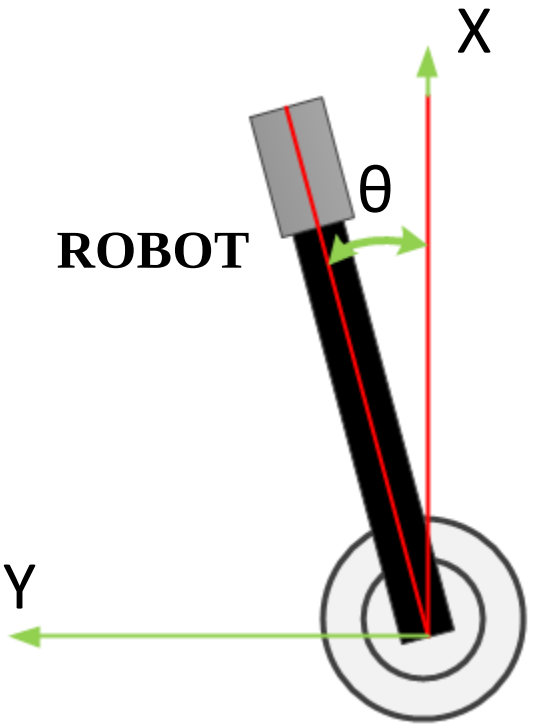
3.1. Osvježavanje informacija o poziciji robota

Prilikom svakog pokretanja programa potrebno je odabrati na kojem je *portu* računalo povezano s kinematičkim kontrolerom. Ako je program preko kinematičkog kontrolera povezan s robotom, pokreće se timer koji traži od kinematičkog kontrolera informaciju o poziciji robota svakih 100 ms. Sustav osvježavanja informacija o poziciji robota radi tako da

svaki put kad zatraži trenutnu poziciju robota podiže zastavicu kojom signalizira dijelu programa koji prima podatke s kinematičkog kontrolera da bi sljedeći podaci koje primi trebali biti za poziciju robota. Pseudokod ovog dijela programa je sljedeći:

```
// Kinematički kontroler je poslao podatke na računalo u varijablu P te zastavica F_xyz
//iznosi 1
begin
 if F_xyz = true then  //zastavica nam govori da bi ovi podaci mogli biti o poziciji
  for i=0 to length(P)-1 do
   if Pozicija = true then Nove_Poz   //podaci sadrže x,y,z,f,p slova, nova pozicija i F_xyz=0
   else Ostali_Podaci;   //podaci ne sadrže x,y,t,f,p slova- drugi podaci i F_xyz = 1
end;
```

Pri osvježavanju informacija ujedno se vrši i računanje kuta zakreta robota. Kut zakreta robota potreban je pri izračunu novih koordinata robota pomoću slike primljene s web-kamere.



Slika 9. Kut zakreta robota

Vrijednosti kuta zakreta računamo prema sljedećim izrazima:

$$x = 0 \quad \begin{matrix} y > 0 \;\rightarrow\; \theta = \frac{\pi}{2} \\ y < 0 \;\rightarrow\; \theta = -\frac{\pi}{2} \end{matrix}$$

$$\theta = \tan^{-1}\frac{Robot.y}{Robot.x}$$

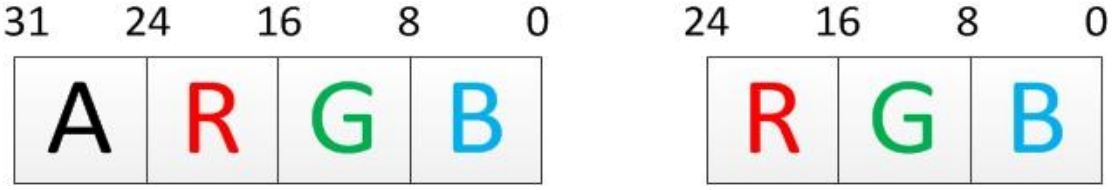
3.2. Uzorkovanje slike

Uzorkovanje slike vrši se pomoću VLSnaphot i TLClockGenerator komponenti. Komponenta TLClockGenerator služi za određivanje frekvencije uzorkovanja koje se vrši na komponenti VLSnaphot. Na njoj je moguće podesiti devet načina okidanja signala koji se razlikuju po prioritetu. Najniži prioritet omogućuje komponenti da radi samo kad je računalo u „nezaposlenom“ načinu rada (*eng. idle mode*) gdje greška može biti veoma velika, dok najviši prioritet rada znači da će komponenta ispravno raditi bez obzira na trenutnu zaposlenost računala. Odabran je normalan način rada gdje greška komponente prilikom uzorkovanja može iznositi maksimalno 1 ms. Maksimalna brzina uzorkovanja je 30 *frameova* u sekundi zbog toga što je to maksimalna brzina snimanja web-kamere. [10]

3.3. Obrada slike

Obrada slike vrši se tako da slika primljena s web-kamere prolazi kroz tri različite strukture filtera od kojih aktivne mogu biti sve tri ili nijedna. Ako nijedna struktura filtera nije aktivna, tada se slika primljena s web-kamere direktno prikazuje korisniku na zaslonu računala.

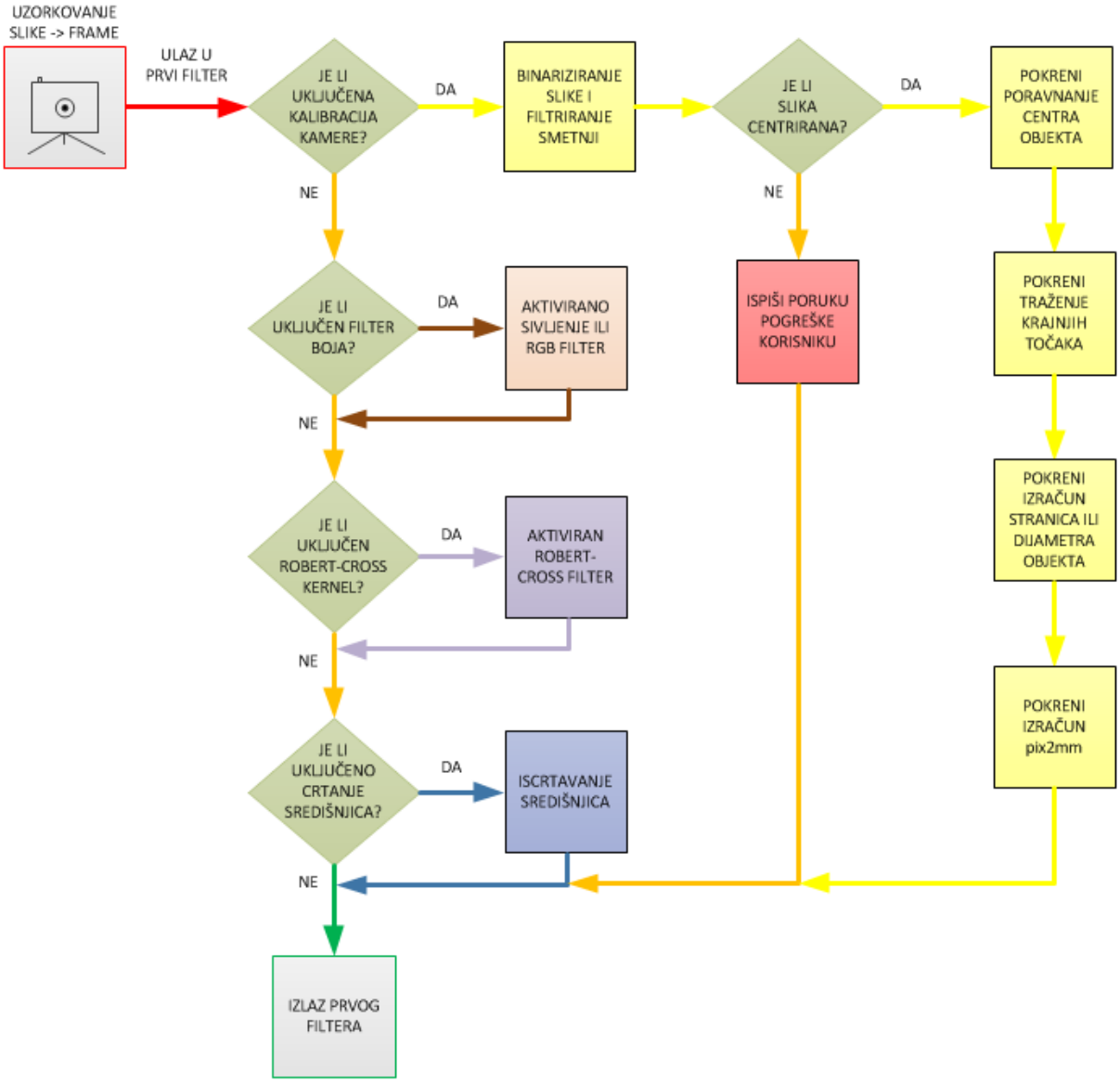
Da bismo mogli obavljati ikakve računalne operacije na prihvaćenoj slici, potrebno je sliku prevesti u računalu razumljiv kod. Prvo je potrebno odabrati s kojim formatom prihvaćene slike želimo raditi. Mogući prihvatljivi formati su 24-bitni i 32-bitni. Oni se razlikuju po veličini i brzini obrade, ali ne i po kvaliteti slike. Slika koja je zapisana u 24-bitnom formatu zauzima manje mjesta u memoriji od one koja je zapisana u 32-bitnom formatu, no zbog toga što je operativni sustav Windows programiran da radi sa 32-bitnim podacima, 24-bitni format je sporiji prilikom učitavanja u memoriju ili iscrtavanja iz nje. Zbog toga što će sustav prepoznavanja raditi u realnom vremenu, odabrani je 32-bitni format slike.



Slika 10. Prikaz zapisa bitova boja kod 32-bitne i 24-bitne slike

3.3.1. Struktura prvog filtera

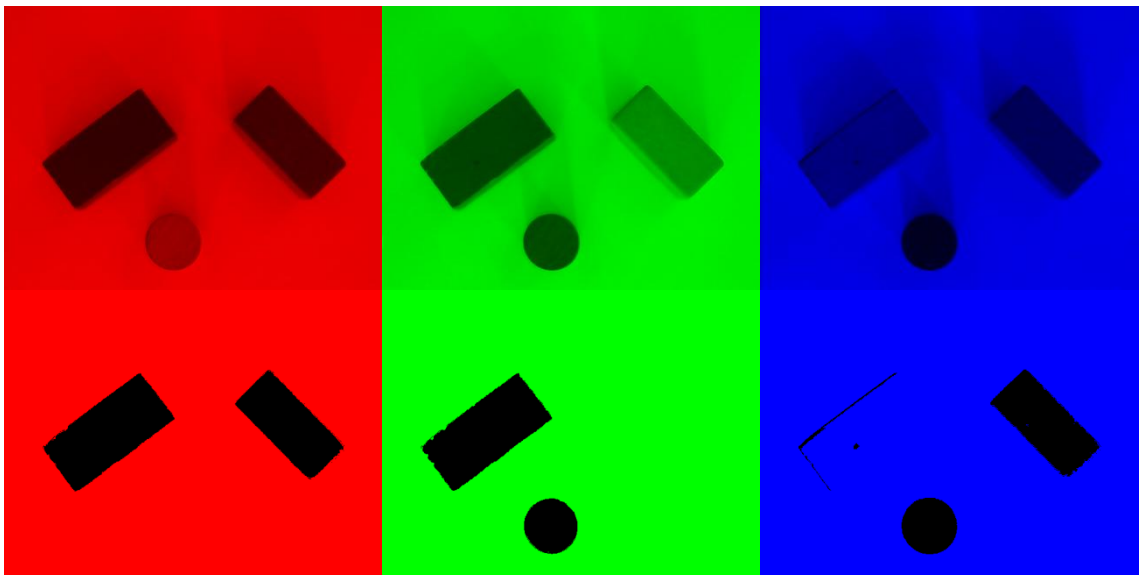
Ulazni parametar strukture prvog filtera je uzorkovana slika koju daje komponenta VLSnaphot. U strukturi prvog filtera mogu se istovremeno odvijati maksimalno tri od četiri zasebna procesa. Ti procesi su kalibracija kamere, filtriranje boja, Robert Cross operator i crtanje pomoćnih središnjica.



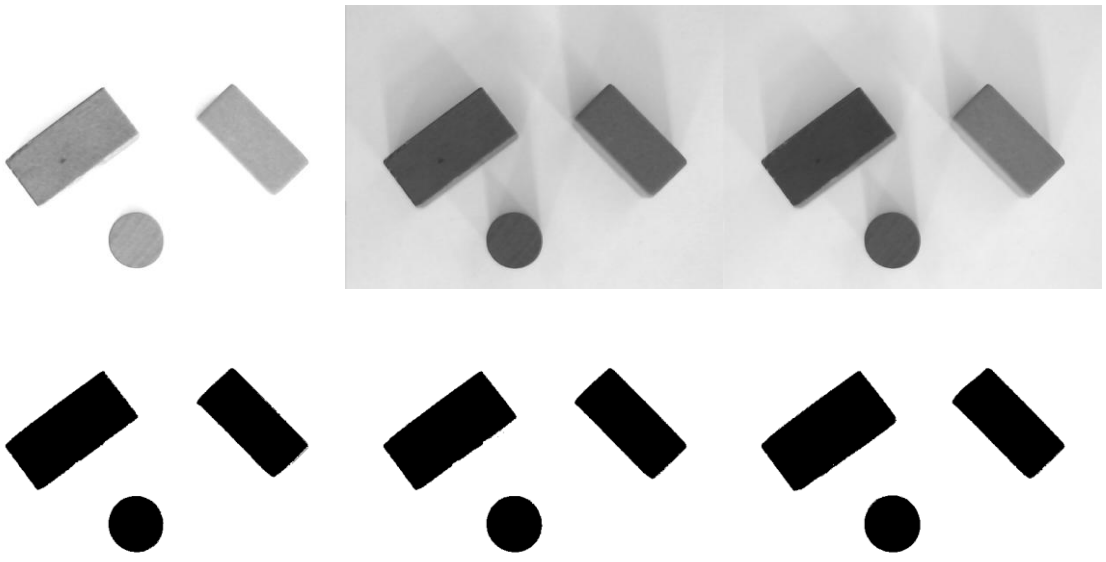
Slika 10. Struktura prvog filtera

3.3.1.1. Filtriranje boja

Prema odabranim postavkama, s grafičkog sučelja programa u ovom se dijelu strukture prvog filtera može vršiti šest različitih modificiranja primljene slike te njena binarizacija. Potrebno je koristiti različite filtere zbog toga što izlazna slika s web-kamere dolazi s mnogo šuma koji je nemoguće filtrirati samo pomoću filtera smetnji.



Slika 11. Pogled kamere kroz RGB filtere bez binarizacije slike i s njom



Slika 12. Pogled kamere kroz filtere sivljenja bez binarizacije slike i s njom

RGB filteri rade na principu da boju koju odaberemo ostave kakvom jest, dok ostale dvije komponente postavlja na vrijednost nula. Ovakvim načinom filtriranja možemo lako

uočiti predmete koji su crvene, zelene ili plave boje te bijelog kontrasta zbog toga što će se boja tih predmeta sačuvati dok će se ostale komponente izgubiti.

Ugrađena funkcija sivljenja može koristiti tri načina stvaranja sive slike iz slike u boji. Ta tri načina su sivljenje po svjetlini (*eng. lightness*), sivljenje po uprosječivanju boje (*eng. average*) te sivljenje po osvjetljenju (*eng. luminosity*). Svaka od ove tri metode ima svoje primjene u određenim uvjetima, stoga korisnik programa mora sam odlučiti koja mu metoda u određenom trenutku najviše odgovara. Sivljenje po svjetlini smanjuje kontrast slike, a računa tako da se zbroje najutjecajnija komponenta boje i najmanje utjecajna komponenta boje te se podjele sa brojem dva. Sivljenje po osrednjenju boja vrši se tako da se zbroje sve tri komponente boje te se zbroj dijeli s brojem tri. Sivljenje po osvjetljenu je najsofisticiranija metoda sivljenja. Ona, uzimajući u obzir ljudsku percepciju boja, ponderira sve tri komponente boje određenim vrijednostima. Ljudsko oko je najosjetljivije na zelenu boju tako da ona pri računanju sivljenja ovdje dobiva najveću vrijednost. Izrazi korišteni za metode sivljenja:

- sivljenje po svjetlini:

$$f\ R,G,B\ = \frac{(max\ R,G,B\ + min\ R,G,B\ )}{2}$$

- sivljenje po uprosječivanju boja:

$$f\ R,G,B\ = \frac{R+G+B}{3}$$

- sivljenje po osvjetljenju:

$$f\ R,G,B\ = 0.2126 \cdot R + 0.7152 \cdot G + 0.0722 \cdot B$$

gdje je: *f(RGB)* – rezultat funkcije sivljenja

R – intenzitet crvene boje

G – intenzitet zelene boje

B – intenzitet plave boje

Maksimalna vrijednost funkcije *f(RGB)* iznosi 255, što odgovara bijeloj boji, a to se događa kada sve tri komponente boje iznose 255. Najmanja vrijednost iznosi 0, što odgovara crnoj boji.

Pseudokod funkcija filtera izgleda ovako:

```
funkcija Filter (ulaz : i,j)   //i,j označavaju lokaciju  trenutnog piksela
 begin
  if RGB = true then   //  nad slikom vršimo RGB filter
   if R = true then      Slika[i,j].GB = 0      else      //crveni filter
   if G = true then      Slika[i,j].RB = 0      else      //zeleni filter
   if B = true then      Slika[i,j].GR = 0      else      //plavi filter
   Slika[i,j].RGB = Sivljenje;      //sivljenje slike
 end;
```

Nakon provođenja RGB filtera ili funkcije sivljenja potrebno je provesti funkciju binariziranja piksela te spremanja slike u matricu. Da bismo odredili koji pikseli na slici odgovaraju objektu a koji pozadini, potrebno je provesti neku vrstu filtera. Odabrani filter je jednostavno ograničavanje maksimalne vrijednosti piksela (granica filtera je promjenjiva unutar programa). Ograničenje radi na principu da sve piksele koji imaju vrijednost veću od 120 proglašava bijelim pikselima i daje im vrijednost nula, dok sve piksele koji imaju vrijednost manju od 120 proglašava crnim te im daje vrijednost jedan. Nakon što je slika binarizirana, njene vrijednosti pohranjuju se u matricu slike čije dimenzije odgovaraju dimenzijama slike (redci su jednaki broju piksela po visini, dok su stupci jednaki broju piksela po širini).

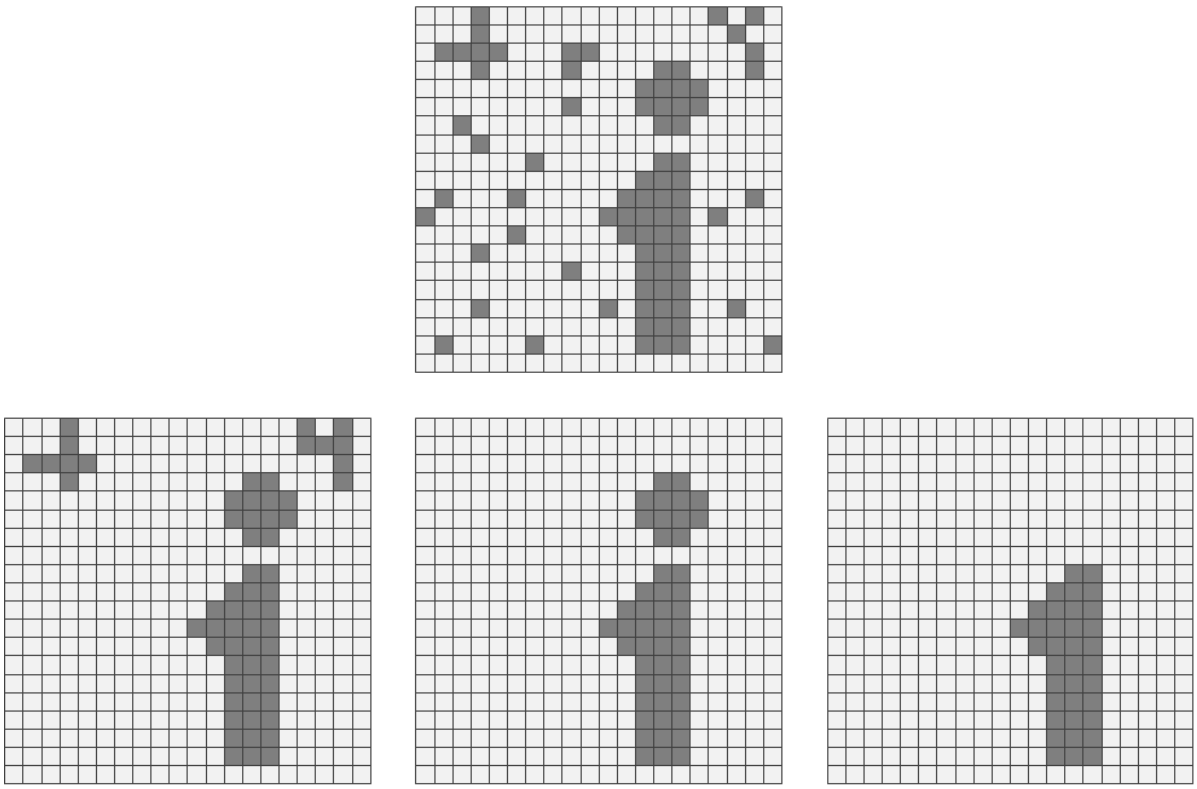
Pseudokod funkcije binariziranja izgleda ovako:

```
for y = 0 to visina(Slika)-1 do
 begin
 S= redak(Slika);   // funkcija koja učitava redak slike
 for x = 0 to širina(Slika)-1 do
  begin
   S[i,j] = Filter[i,j];   // RGB ili sivljenje
   if binariziranje = true then
    if S > granica then   // Binariziranje piksela
     Matrica[i,j] = 0    else
     Matrica[i,j] = 1;
  end;  end;
```

3.3.1.2. Kalibracija kamere

Prilikom pokretanja programa potrebno je pokrenuti kalibraciju kamere. Uspješnom kalibracijom kamere dobiva se omjer koliko jedan piksel na primljenoj virtualnoj slici iznosi milimetara u stvarnosti. Poznavanje tog omjera potrebno je za uspješno pomicanje robota prema objektima prepoznatima na slici. Nadalje, da bismo uspješno pokrenuli kalibraciju kamere, potreban preduvjet je da smo ranije aktivirali barem jedan od ponuđenih filtera te da smo aktivirali proces binarizacije slike.

Prije ikakvog prepoznavanje objekata ili kalibracije kamere, primljenu sliku potrebno je provesti kroz filter smetnji. Filter smetnji koristi se zbog toga što mali objekti na binarnoj slici nisu objekti, već nepoželjan šum koji ne sadrži nikakvu korisnu informaciju. Da bismo uklonili šum s binarne slike koristimo filter veličine objekta kao filter smetnji. Filter veličine objekta radi na principu da svaki pronađeni objekt koji ima veličinu manju od zadane proglašavamo nevažećim, tj. piksele koji ga označavaju izjednačavamo s pikselima pozadine. Problem ovog filtera jest u tome što je teško pronaći dobru zadanu veličinu objekta, a kad je i pronađemo, ona vrijedi samo u određenim uvjetima. Prilikom izrade rada dogodilo se da granica pronađena za prihvatljivo filtriranje objekata ujutro nije bila ispravna poslijepodne tijekom istog dana. Ako zadanu granicu objekta postavimo prenisko, nećemo odstraniti sav šum sa slike, ali ako zadanu granicu postavimo previsoko, tada možemo izgubiti korisne informacije.



Slika 13. Primjeri premale, dobre i prevelike granice filtera veličine objekta

Kako bismo mogli koristiti filter veličine objekta, prvo je potrebno svakoj grupi piksela na slici dodijeliti određeni identifikacijski broj u matrici slike. Ova metoda slijedi sljedeća pravila:

1) Kreni pregledavati sliku od lijeva na desno, od vrha prema dnu.

2) Ako je vrijednost piksela jednaka nuli (jedan je vrijednost pozadine), tada:

   a) ako samo jedan od piksela koji se nalaze iznad trenutnog piksela i lijevo od njega ima oznaku, tada kopiraj tu oznaku na ovaj piksel;

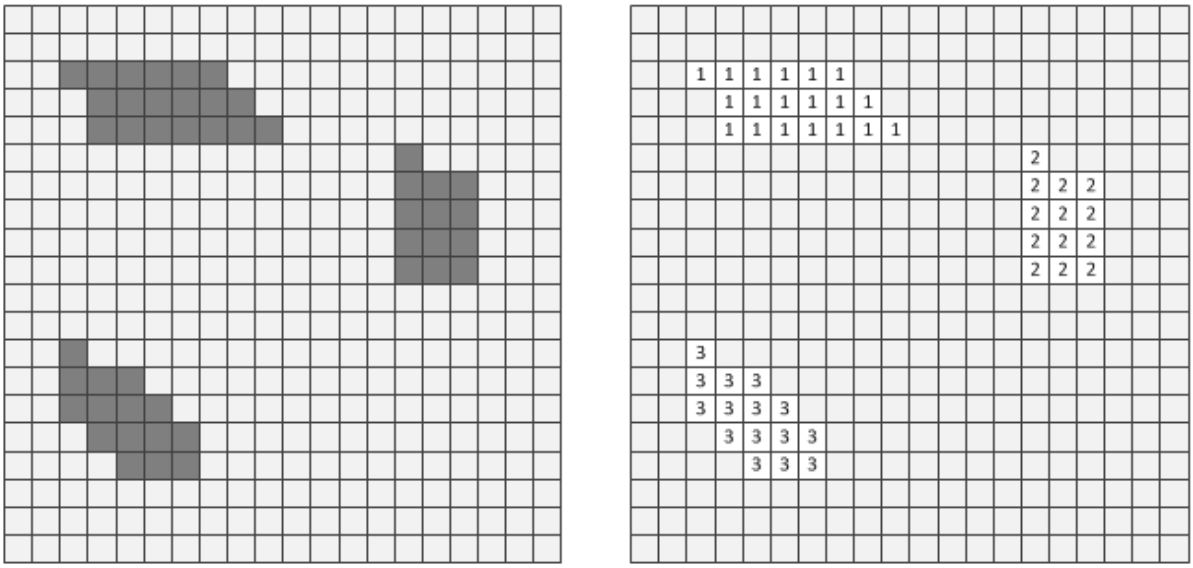
   b) ako oba piksela koja se nalaze iznad trenutnog piksela i lijevo od njega imaju istu oznaku, tada kopiraj tu oznaku na ovaj piksel;

   c) ako pikseli koji se nalaze iznad trenutnog piksela i lijevo od njega imaju različite oznake, tada kopiraj oznaku piksela iznad sebe kao svoju oznaku te upiši svoju oznaku i oznaku piksela lijevo od sebe u skup za mijenjanje;

d) ako nijedan od piksela koji se nalaze iznad trenutnog piksela i lijevo od njega nema oznaku, tada inkrementiraj registar oznaka i dodijeli tu novu oznaku na ovom pikselu.

3) Ako u skupu za mijenjanje postoji oznaka, tada kreni u matrici slike od vrha prema dnu, od lijevo prema desno i zamijeni tu oznaku zamjenskom oznakom.

4) Ako je desno od tebe piksel vrijednosti nula, tada mu dodijeli svoju vrijednost, a ako nije, vrati se na korak 1) dok ne prođeš cijelu sliku.



Slika 14. Dodjeljivanje identifikacijskih brojeva objektima u matrici slike

Nakon uspješno provedene filtracije slike za objektima manjima od zadane granice u procesu kalibracije kamere slijedi proces provjeravanja centra kamere. Bitno je provjeriti nalazi li se kamera iznad kalibracijskog objekta ili pokraj njega. Ako se ne nalazi iznad, program će ispisati poruku korisniku da je potrebno kalibracijski objekt postaviti u sredinu kamere ili robot s kamerom pomaknuti iznad objekta. Ako se kamera nalazi iznad kalibracijskog objekta, tada se pokreće procedura traženja centra kalibracijskog objekta. Ova procedura potrebna je zbog toga što su tijekom izrade ovog rada bile implementirane razne metode lociranja krajnjih točaka objekata koje nisu radile ispravno ako je početno pretpostavljeno središte objekta bilo postavljeno daleko od stvarnog središta objekta, i zbog

toga što kamera nije postavljena savršeno okomito na površinu gdje tražimo objekte pa slika koju ona prosljeđuje računalu nije ista u svim točkama. Kao točka s najmanje izobličenja uzeta je sredina kamere te se zbog toga traži da se središte kamere nalazi na kalibracijskom objektu.

Ako se kalibracijski objekt nalazi u središtu kamere, pokreće se procedura za poravnanjem središta objekta. Ova procedura koristi princip traženja točaka u osam smjerova s ishodištem u središtu kamere kako bismo pronašli koordinate točki objekta. Pronađene koordinate točaka pretvaramo u udaljenosti od središta kamere te na taj način možemo znati nalazi li se pretpostavljeno središte kamere blizu ili daleko, iznad pravog središta objekta ili ispod njega. Nakon što smo izračunali udaljenosti, potrebno je te udaljenosti prevesti u nove koordinate središta kalibracijskog objekta. Nove koordinate središta kalibracijskog objekta možemo izračunati na dva načina te se u oba načina računanja od osam pronađenih krajnjih točaka objekta koriste vrijednosti njih šest zato što za računanje pomaka po x-osi ne možemo korisiti dvije točke koje imaju istu vrijednost x, te isto vrijedi i za y-os. Prvi način uzima vrijednosti udaljenosti x-koordinata krajnjih točaka koje se nalaze desno od središta kamere i oduzima ih od onih koje se nalaze lijevo od nje, te uzima vrijednosti udaljenosti y-koordinata krajnjih točaka koje se nalaze iznad središta kamere i oduzima ih od onih koje se nalaze ispod nje. Nakon što dobijemo razliku udaljenosti u smjeru x i y, dijelimo je s tri i pridodajemo vrijednosti središta kamere, te smo time dobili novo središte objekta. Problem ove metode jest što vrijednosti novog središta objekta imaju prebačaj nad stvarnim središtem objekta. Drugi način uzima isto tako vrijednosti udaljenosti koordinata x i y kao i prvi način, ali umjesto da odmah oduzmemo vrijednosti udaljenosti, ovdje udaljenosti prvo kvadriramo. Nakon kvadriranja i oduzimanja vrijednosti ćemo korjenovati i ovisno o tome je li član ispod korijena bio pozitivan ili negativan, pribrojati ili oduzeti od vrijednosti središta kamere kako bismo dobili novo središte objekta. Ova metoda daje mnogo bolje rezultate od prve metode jer ona ne daje rezulate s prebačajima preko stvarnog središta objekata, već se samo približavamo njemu.

Pseudokod poravnanja središta objekta je slijedeći:

```
S= redak(Kamera_y)    // funkcija koja učitava redak slike
x = Kamera.x   //  početna točka je središte kamere
While (objekt) do
  begin
    Desna_tocka=[x,y];    inc(x);    //traženje desne točke
  end;
x = Kamera.x;
While (objekt) do
  begin
    Lijeva_tocka=[x,y];    dec(x);    //traženje lijeve točke
  end;
[x,x1,x2] = Kamera.x
While  (Objekt = true)  do
 begin
   S= redak(Kamera_y);    // funkcija koja učitava redak slike
   if Objekt[x-1] = true then GornjaLijevo_tocka = [x1-1,y]    //traženje gornje-lijevo točke
   if Objekt[x] = true then Gornja_tocka = [x,y]    //traženje gornje točke
   if Objekt[x+1] = true then GornjaDesno_tocka = [x2+1,y]    //traženje gornje-desno točke
  inc[y,x2];  dec(x1);
 end;
y = Kamera.y
[x,x1,x2] = Kamera.x
While  (Objekt = true)  do
 begin
   S= redak(Kamera_y);    // funkcija koja učitava redak slike
   if Objekt[x-1] = true then DonjaLijevo_tocka = [x1-1,y]    //traženje donje-lijevo točke
   if Objekt[x] = true then Donja_tocka = [x,y]    //traženje donje točke
   if Objekt[x+1] = true then DonjaDesno_tocka = [x2+1,y]    //traženje donje-desne točke
  inc(x2);  dec[y,x1];
 end;
```

Formule korištene za prvi način izračuna novog središta:

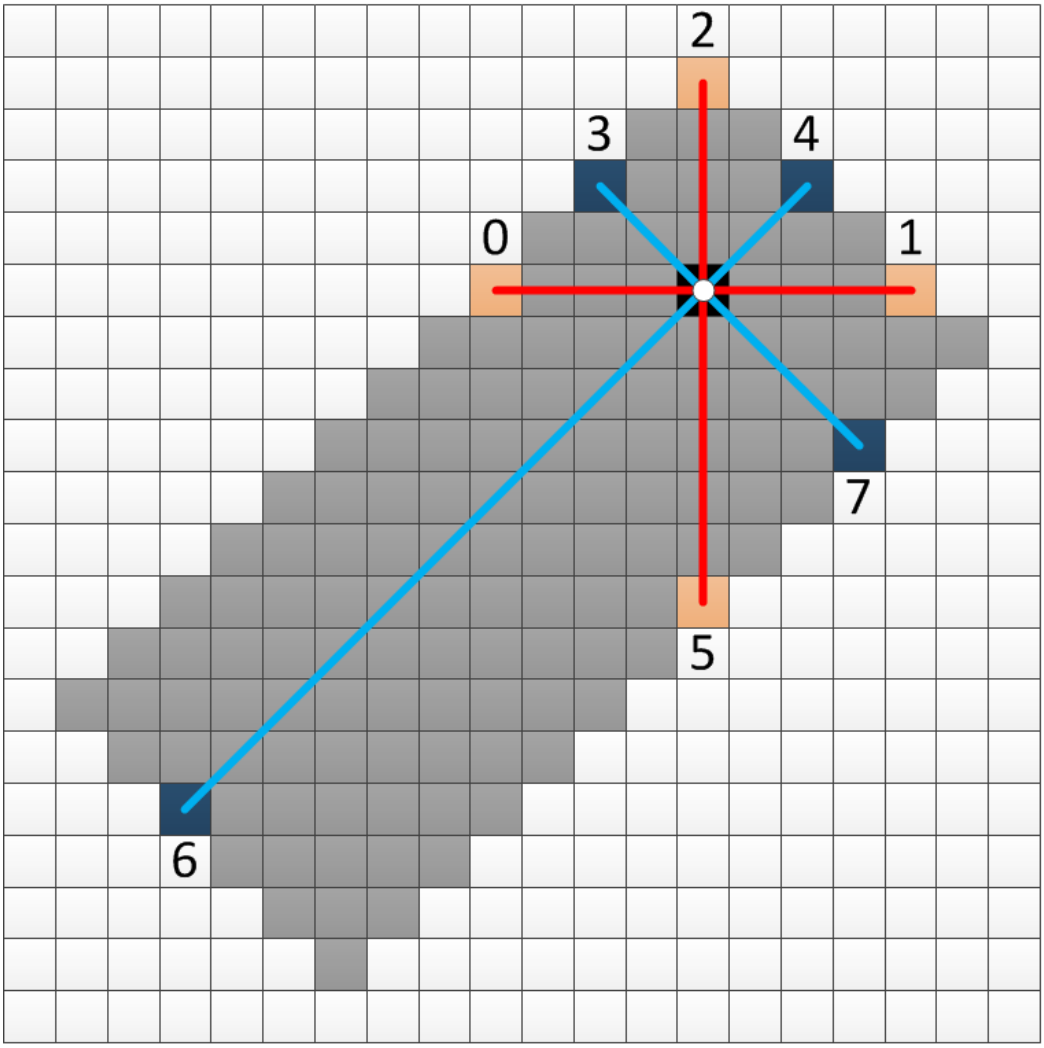
$$X = X_{sredista} + \frac{T_{1x} + T_{4x} + T_{7x} - T_{0x} - T_{3x} - T_{6x}}{3}$$

$$Y = Y_{sredista} + \frac{T_{2x} + T_{3x} + T_{4x} - T_{5x} - T_{6x} - T_{7x}}{3}$$

Formule korištene za drugi način izračuna novog središta:

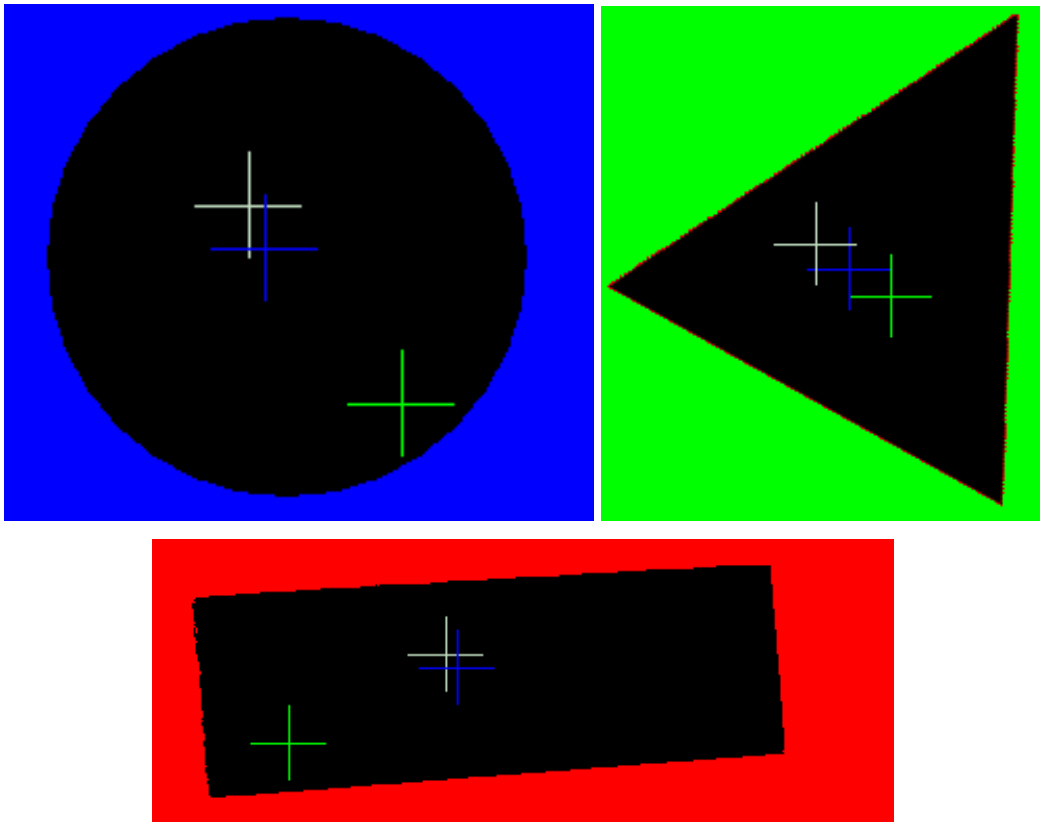
$$X = X_{sredista} \pm \sqrt{T_{1x}^2 + T_{4x}^2 + T_{7x}^2 - T_{0x}^2 - T_{3x}^2 - T_{6x}^2}$$

$$Y = Y_{sredista} \pm \sqrt{T_{2x}^2 + T_{3x}^2 + T_{4x}^2 - T_{5x}^2 - T_{6x}^2 - T_{7x}^2}$$



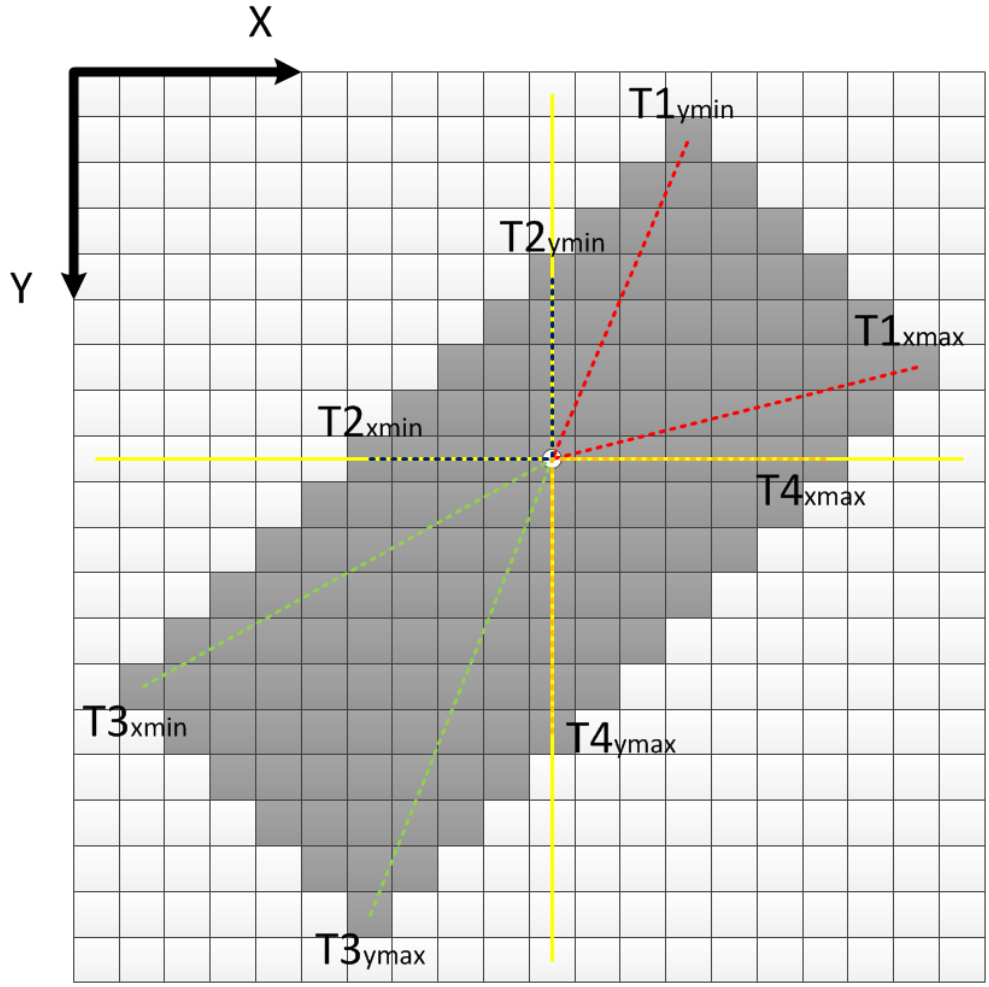
Slika 15. Traženje točaka objekta pomoću metode osmosmjernog pretraživanja

Slika 16 prikazuje nova središta objekata pronađena metodom osmosmjernog pretraživanja. Na slici je zelenim križićem prikazano središte kamere, sivim križićem rezultat dobiven prvom metodom, a plavim križićem rezultat dobiven drugom metodom.



Slika 16. Nova središta objekata pronađena metodom osmosmjernog pretraživanja

Nakon što smo uspješno proveli proceduru poravnanja središta objekata, pokrećemo proceduru traženja krajnjih točaka objekta. Traženjem krajnjih točaka objekta želimo pronaći četiri koordinate na slici koje odgovaraju najisturenijim točkama – najvišoj, najnižoj, najljevljoj i najdesnijoj točki objekta. Tijekom izrade rada ovdje su korištene dvije metode. Prva metoda je metoda kvadratnog pretraživanja koja traži maksimalne i minimalne točke unutar svog kvadranta. Ova metoda daje dobre rezultate, ali ima kompliciran računalni kod koji zahtijeva mnogo resursa. Druga metoda je metoda pretraživanja unutar određenih granica. Ova metoda daje čak i bolje rezultate od prve metode i ima mnogo jednostavniji računalni kod za izvršavanje.

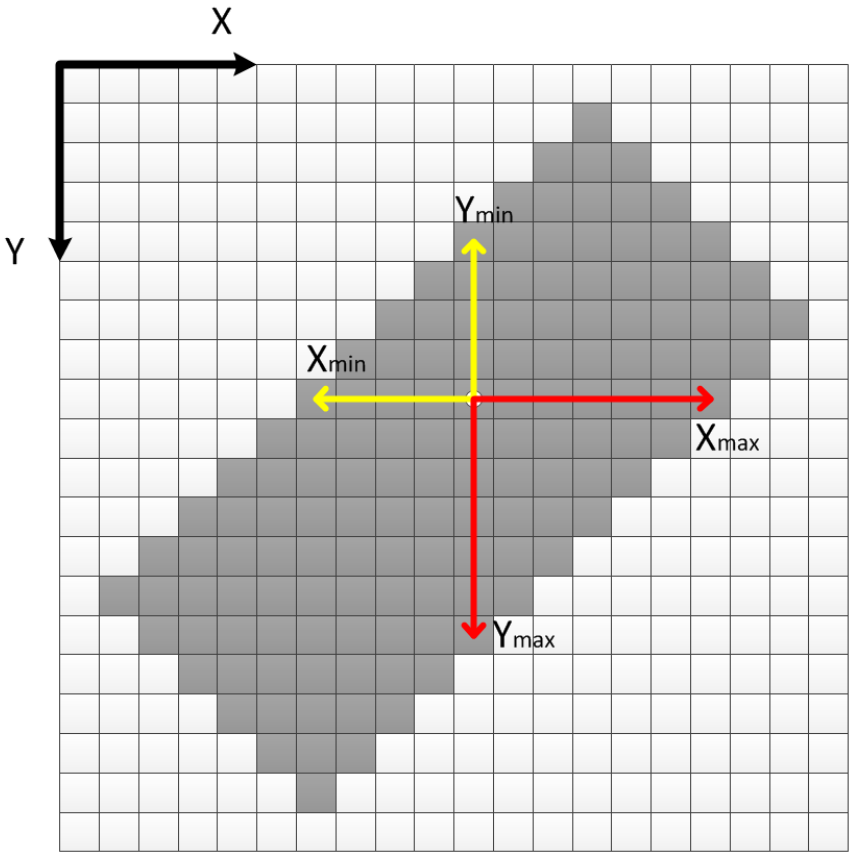


Slika 17. Metoda kvadratnog pretraživanja

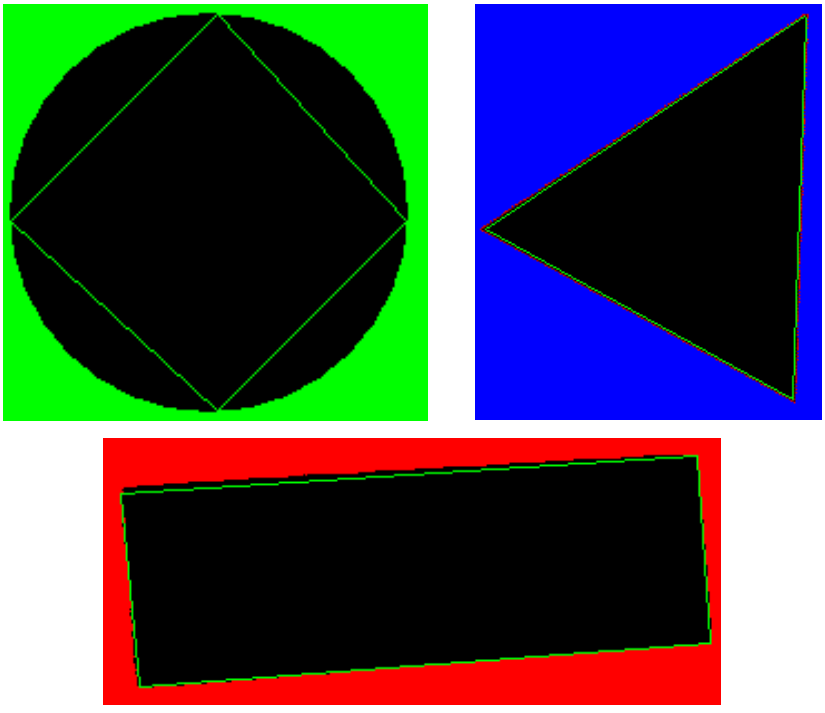
Metoda kvadratnog pretraživanja radi na principu dijeljenja slike na četiri kvadranta te posebnog pretraživanja svakog kvadrata krećući od prvog do četvrtog za minimalnim ili maksimalnim vrijedostima. Rezultat pretraživanja svakog kvadranta bit će dvije točke koje će označavati ekstreme u tom kvadrantu. U prvom kvadrantu traže se točke objekta s maksimalnom vrijednosti x i minimalnom vrijednosti y. U drugom kvadrantu traže se točke sa minimalnom vrijedosti x i y. U trećem kvadrantu traže se točke sa minimalnom vrijednosti x i makismalnom vrijednosti y. U četvrtom kvadranut traže se točke s maksimalnom vrijednosti x i y. Nakon što je algoritam pretražio područje cijele slike i pronašao osam točaka, potrebno je odlučiti koje od tih osam točaka su zaista ekstremi tog objekta. Pronalaženje ekstrema radi se tako da se prolazi kroz pronađene točke s uvjetima maksimalne i minimalne vrijedosti komponenti x i y. Ovaj algoritam vrlo je ovisan o početnom pretpostavljenom položaju središta. Ako se pretpostavljeno središte jako razlikuje od stvarnog središta, metodi su potrebne dodatne podfunkcije kako bi ispravno radio, što pogoršava njegove performanse i povećava složenost programskog koda.

Metoda pretraživanja unutar određenih granica radi na principu postavljanja pretpostavljenih granica unutar kojih se nalazi objekt te pretraživanja samo unutar tih granica. Ova metoda može davati veoma loše rezultate ako se koristi samostalno bez procedure za poravnanjem središta objekta. S procedurom za poravnanjem središta objekta daje zadovoljavajuće rezultate pomoću kojih se objekti mogu prepoznavati te je njen programski kod kompaktan. Prije početka pretraživanja potrebno je postaviti granice pretraživanja. Granice se postavljaju tako da se pronađu udaljenosti između maksimalnih i minimalnih vrijednosti x i y sa središtem u središtu kamere. Tada se te udaljenosti zbroje i podijele sa vrijednosti 2,5. Kada smo pronašli granice pretraživanja, tada može započeti pretraživanje slike za maksimalnim i minimalnim vrijednostima objekta. Pseudokod metode pretraživanja izgleda ovako:

```
S= redak(Slika.srediste_kamere);    // funkcija koja učitava redak slike
While (objekt = true) do
  begin
    Pronađi(max_x)   // tražimo maksimalne
    Pronađi(min_x)   // i minimalne
    Pronađi(max_y)  // vrijednosti objekta
    Pronađi(min_y)
  end;
  Limit = ((max_x – min_x) + (max_y – min_y)) / 2
for y = (Središte_kamere - limit) to (Središte_kamere + limit) do
  begin
   S= redak(Slika);    // funkcija koja učitava redak slike
   for x = (Središte_kamere - limit) to (Središte_kamere + limit) do
    begin
       if Objekt = true  then  Test_krajnja_točka
    end;  end;
```



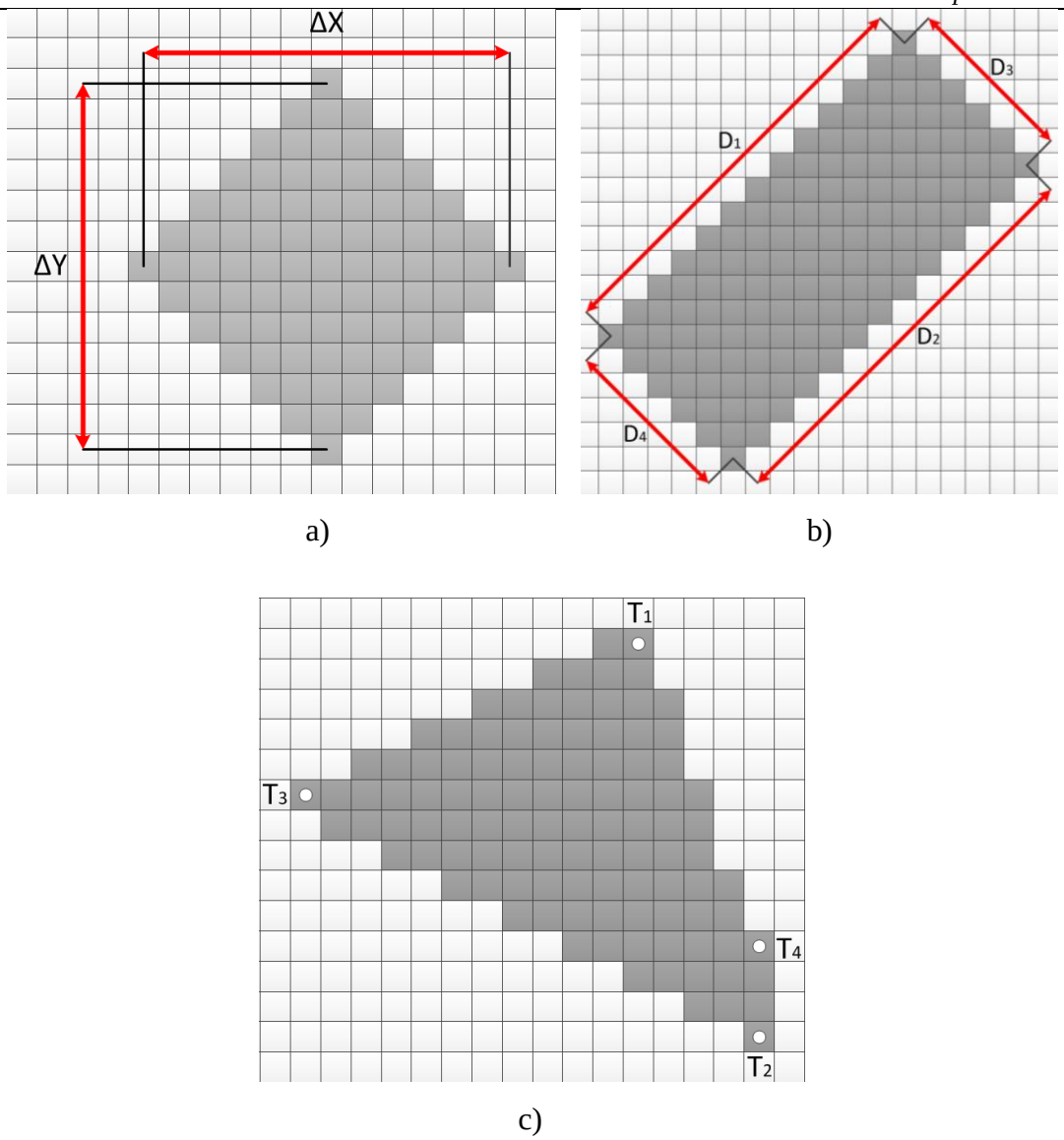
Slika 18. Način pronalaženja granica kod metode pretraživanja unutar određenih granica



Slika 19. Krajnje točke objekta pronađene pomoću metode pretraživanja unutar određenih granica

Nakon što smo uspješno proveli metodu pretraživanja unutar određenih granica i kao rezultat dobili krajnje točke objekta, možemo pokrenuti proceduru za izračunom stranica ili dijametra objekta. Osim što ćemo ovom procedurom dobiti dimenzije objekta, ujedno ćemo dobiti informaciju je li pronađeni objekt četverokut, trokut ili krug. Ovom procedurom dobit ćemo šest podatka o predmetu, a to su udaljenosti između krajnje točke koja označava najmanji y te maksimalni i minimalni x, udaljenost između krajnje točke koje označava maksimalni y te minimalnog i maksimalog x, udaljenosti između minimalnom i maksimalnog y, te udaljenosti između minimalnog i maksimalnog x. Prve četiri dimenzije su stranice objekta, dok su preostale dvije dimenzije dijagonale objekta. Ako je razlika udaljenosti između maksimalne i minimalne vrijednosti x te maksimalne i minimalne vrijednosti y približno jednaka, možemo zaključiti da je pretraživani objekt krug. Međutim, ta razlika nikada neće biti jednaka zbog loše kvalitete slike koju šalje web-kamera i zbog kosog pogleda kamere na pretraživani objekt, tj. razlikuje se otprilike pet posto. Nadalje, ako postoji mala razlika između parova udaljenosti, tada se radi o kvadratu, a ako postoje tri isturene točke od kojih se jedna nalazi između te tri točke – nije ekstrem, tada se radi o trokutu. Ako objekt ne ispunjava ni jedno od navedenih pravila tada se radi o nepoznatom objektu.

Slika 20. prikazuje metode prepoznavanja objekata pomoću udaljenosti. Slika a) prikazuje krug koji možemo prepoznati ako su razlike udaljenosti maksimalnih i minimalnih vrijednosti x i y bliske tj. $\Delta X \approx \Delta Y$. Slika b) prikazuje kvadrat koji možemo prepoznati ako su parovi udaljenosti bliski tj. $D_1 \approx D_2$ i $D_3 \approx D_4$ ili $D_1 \approx D_2 \approx D_3 \approx D_4$. Slika c) prikazuje trokut koji možemo prepoznati ako jedna od pronađenih krajnjih točaka nije ekstrem tog objekta. Na slici točka T_1 označava minimum visine objekta, točka T_3 označava minimum širine objekta, točka T_2 označava maksimum visine objekta, dok točka T_4 označava samo jednu točku objekta – nije ekstrem. Zbog toga što imamo tri karakteristične točke umjesto četiri, možemo zaključiti da se radi o trokutu.



a) b)

c)

Slika 20. Korištenje udaljenosti kao metodu prepoznavanja objekata

Udaljenosti karakterističnih točaka računaju se prema sljedećim izrazima:

$$D_1 = \sqrt{T_{1_x} - T_{3_x}\ ^2 + \ T_{3_y} - T_{1_y}\ ^2}$$

$$D_2 = \sqrt{T_{4_x} - T_{2_x}\ ^2 + \ T_{2_y} - T_{4_y}\ ^2}$$

$$D_3 = \sqrt{T_{4_x} - T_{1_x}\ ^2 + \ T_{4_y} - T_{1_y}\ ^2}$$

$$D_4 = \sqrt{T_{2_x} - T_{3_x}\ ^2 + \ T_{2_y} - T_{3_y}\ ^2}$$

$$D_x = \sqrt{T_{1_x} - T_{2_x}\ ^2 + \ T_{2_y} - T_{1_y}\ ^2}$$

$$D_y = \sqrt{T_{4_x} - T_{3_x}\ ^2 + \ T_{4_y} - T_{3_y}\ ^2}$$

Podaci o kalibracijskim objektima nalaze se izvan izvršnog programa u tekstualnoj datoteci naziva „*CPredmet.txt*“. Svi predmeti sa svojim značajkama i dimenzijama učitavaju se u izvršni program prilikom njegova pokretanja. Isto tako, ako se datoteka s kalibracijskim objektima promijeni u toku rada programa, moguće ju je ponovno učitati. Nakon što smo uspješno proveli izračun dimenzija objekta potrebno je dimenzije prepoznatog objekta usporediti s dimenzijama odabranog kalibracijskog objekta. Ako je prepoznati kalibracijski objekt bio krug, tada se uspoređuje dobiven dijametar objekta u pikselima s potrebnom poznatom vrijednošću kruga iz datoteke o kalibracijskim predmetima. Ako je prepoznati kalibracijski objekt bio kvadrat, tada se uspoređuju parovi stranica s prepoznatim stranicama, omjeri piksela se zbrajaju i dijele s brojem dva.

3.3.1.3. Robert Cross operator

Prilikom izrade rada sam u potrazi za što bržim i boljim algoritmima za računalni vid testirao različite algoritme koje se koriste prilikom obrade slika. Algoritmi koje sam testirao su Robert Cross, Sobel, Prewitt i Canny operatori. Od ova četiri operatora odlučio sam ugraditi Rober Cross operator u sustav za prepoznavanje zbog toga što on ima najjednostavniju implementaciju.

Robert Cross operator prvi je operator koji se koristio pri detektiranju rubova na slici. On je diferencijalni operator čija je ideja aproksimiranje gradijenta slike kroz diskretne diferencijale koje izračunavamo sumiranjem razlika dijagonalno razmaknutih piksela.

Rezultantna slika dobija se pomoću sljedećih izraza:

$$G_x = \begin{matrix} +1 & 0 \\ 0 & -1 \end{matrix}$$

$$G_y = \begin{matrix} 0 & +1 \\ -1 & 0 \end{matrix}$$

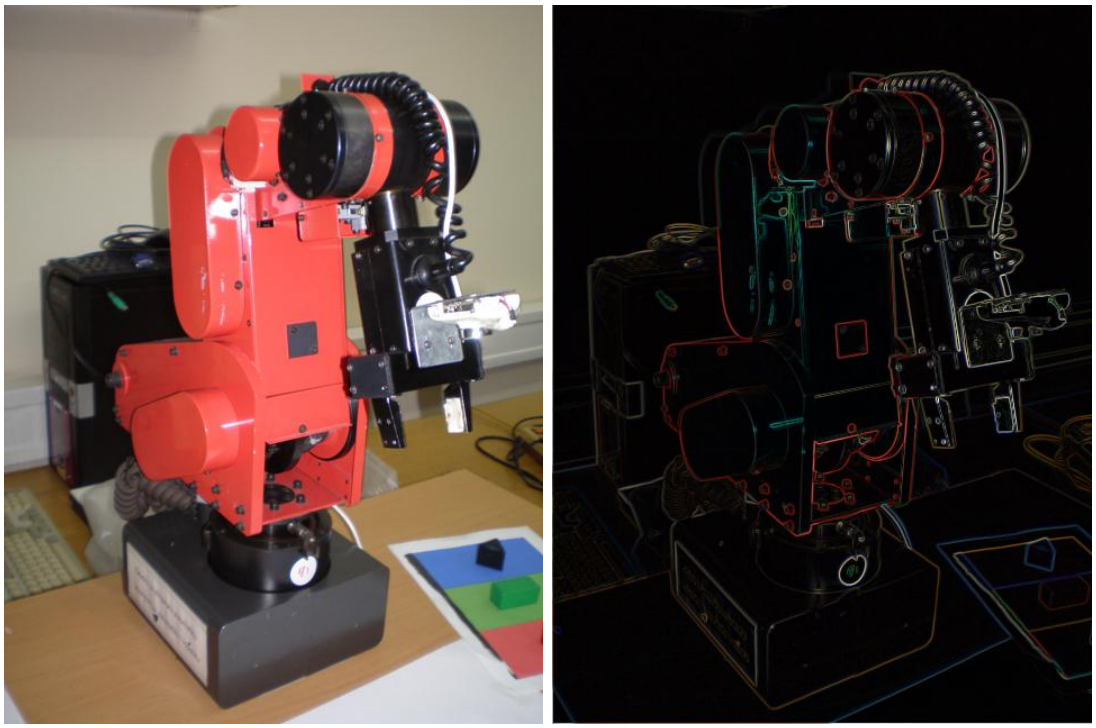
$$\nabla I\ x,y\ = \sqrt{G_x^2 + G_y^2}$$

$$\theta\ x,y\ = \arctan\ \frac{G_y(x,y)}{G_x(x,y)}$$

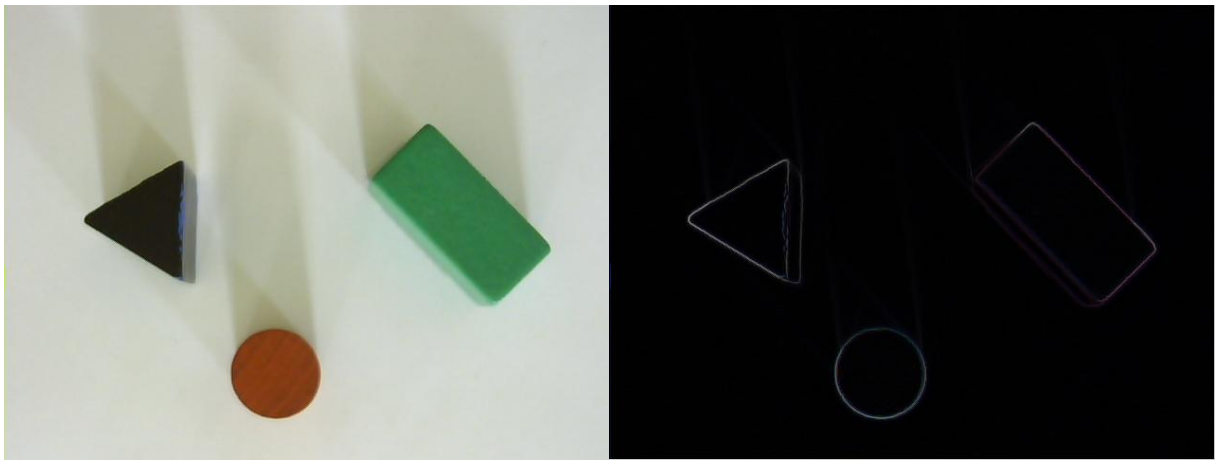
gdje su: *Gx i Gy* – kerneli kojima se vrši konvolucija

$\nabla I\ x,y\ $ – gradijent u točki x,y

$\theta\ x,y\ $ – smjer gradijenta

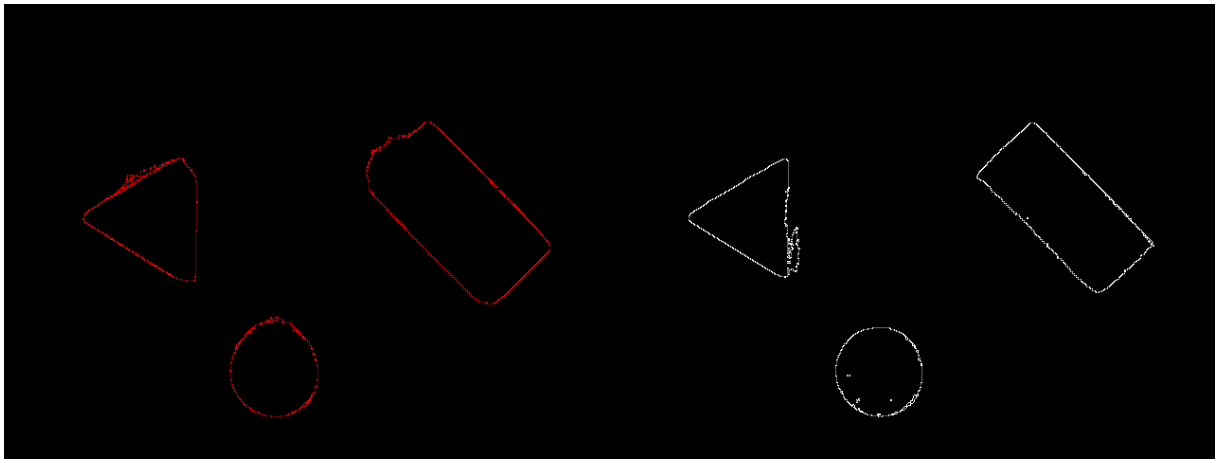


Slika 21. Robert Cross operator



Slika 22. Prepoznavanje objekata pomoću Robert Cross operatora bez filtera

Tijekom izrade rada odlučio sam da obradu slike ne vršim pomoću Robert Cross operatora zato što nisam uspio riješiti problem preosjetljivosti rezultantne slike na sjene koje mogu dodati imaginarne bridove na sliku - kao što je to vidljivo na Slika 22., ili mogu u potpunosti izvitoperiti stranice objekta koji se pokušava prepoznati - kao što je to vidljivo na Slika 23.



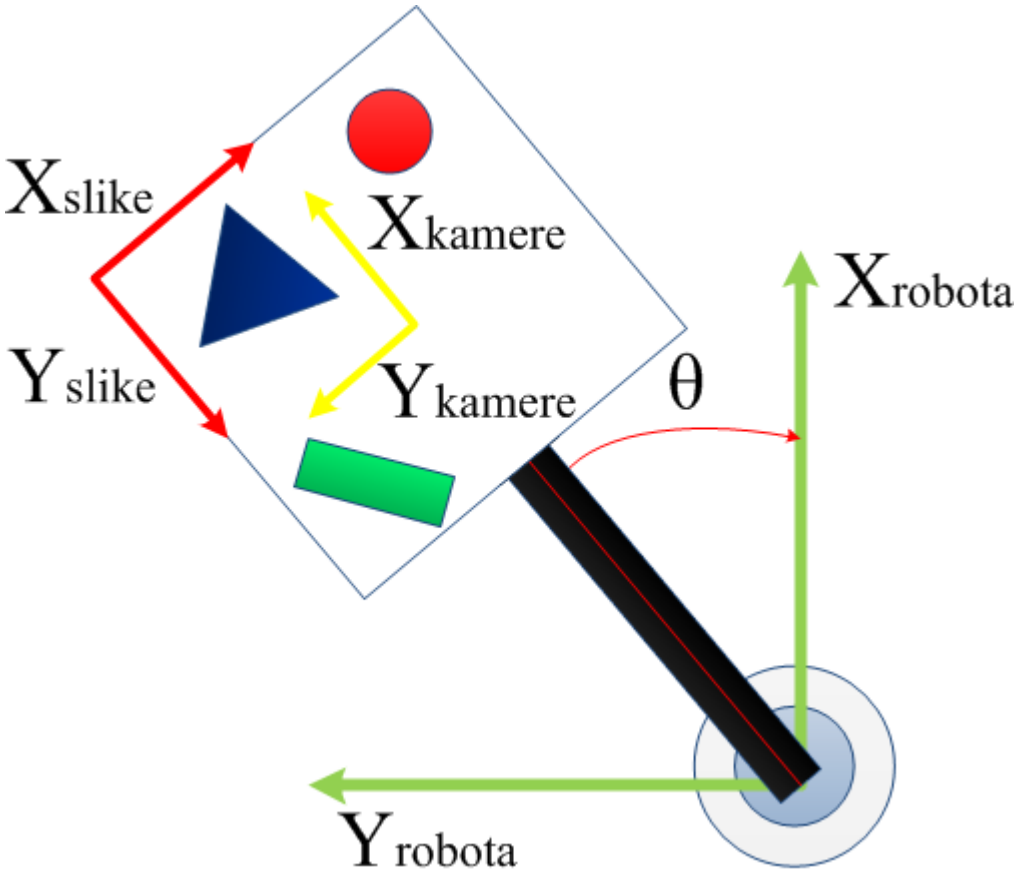
Slika 23. Prepoznavanje objekata pomoću Robert Cross operatora s filterima

3.3.1.4. Crtanje središnjica

Zbog toga što je vrlo teško pogoditi gdje se nalazi centar kamere kada upravljamo robotom i pokušavamo ga dovesti iznad nekog objekta, moguće je aktivirati opciju crtanja središnjica da lakše navigiramo robotom.

3.3.1.5. Transformacija koordinatnog sustava kamere

Ako želimo slati naredbe robotu da se pomiče pomoću informacija o poziciji dobivenih s kamere, potrebno je transformirati kordinatni sustav kamere u koordinatni sustav robota kako bi dobivene informacije bile jednoznačne.



Slika 24. Koordinatni sustavi slike kamere

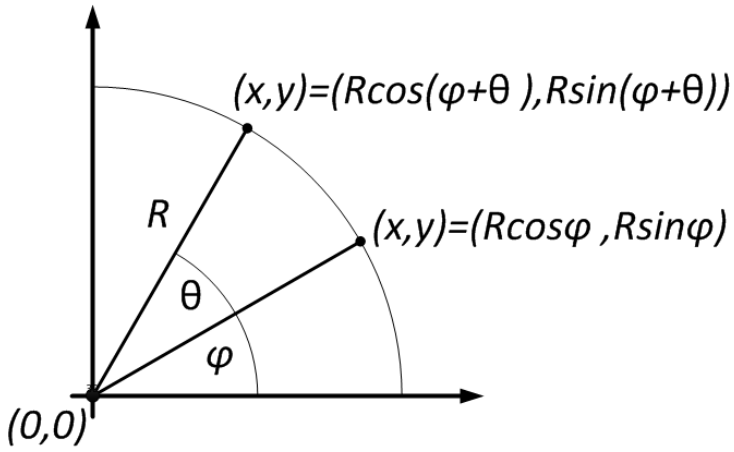
Svaka slika koju primimo s web-kamere ima svoje ishodište u lijevom gornjem kutu. Koordinatna os x ima pozitivan smjer u smjeru širine slike, dok koordinatna os y ima pozitivan smjer u smjeru visine slike. Prvi korak pri prebacivanju koordinatnog sustava slike u koordinatni sustav robota jest da izvršimo pretvorbu željenih koordinata iz koordinatnog sustava slike u koordinatni sustav kamere. Ovo je potrebno jer se programski kod izvršava

tako da će se ono što se nalazi u centru kamere prilikom spuštanja robota naći u centru prihvatnice. Izrazi za pretvorbu koordinata su sljedeći:

$$X_{kamera} = Y_{slike} - Visina_{slike}$$

$$Y_{kamera} = X_{slike} - Širina_{slike}$$

Nakon što smo izvršili pretvorbu koordinata, potrebno je izvršiti dvodimenzionalnu rotaciju oko ishodišta. Dvodimenzionalna rotacija oko ishodišta vrši se na sljedeći način:



Slika 25. Dvodimenzionalna rotacija oko ishodišta

$$x' = R \cdot \cos\ \varphi + \theta$$

$$x' = R \cdot\ \cos\varphi\cos\theta - \sin\varphi\sin\theta\ \ x' = x \cdot \cos\theta - y \cdot \sin\theta\ y' = R \cdot \sin\ \varphi + \theta$$

$$y' = R \cdot\ \sin\varphi\cos\theta + \cos\varphi\sin\theta\ \ y' = y \cdot \cos\theta + x \cdot \sin\theta$$

$$x', y' = x, y \begin{matrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{matrix}$$

$$\begin{matrix} x' \\ y' \end{matrix} = \begin{matrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \end{matrix}$$

gdje je: x, y – koordinate prije rotacije koordinatnog sustava

x', y' – koordinate nakon rotacije koordinatnog sustava

θ – kut rotacije koordinatnog sustava

Ove transformacije koriste se svaki put kada želimo dobiti koordinate prepoznatog objekta na slici kamere u koordinatnom sustavu robota.

3.3.1.6. Prepoznavanja boje objekata

Jedan od zadataka u sklopu izrade ovog rada je i prepoznavanje boje objekata na slici kamere. Kao metode prepoznavanja boja razmatrao sam dvije metode. Prva metoda je RGB metoda koja radi samo ono što je predviđeno ovim zadatkom a to je prepoznavanje crvene, zelene i plave boje, dok je druga metoda prepoznavanja boja – HSV koja može raditi s prepoznavanjem bilo koje odabrane boje.

3.3.1.6.1. RGB metoda prepoznavanja boja

U bilo kojoj tehnici koja šalje svjetlosne signale u boji kao što su zasloni računala ili televizori koristi se tehnika sumacijskog učinka primarnih boja – crvene, zelene i plave. Svaka od tih primarnih boja stimulira ljudske vidne receptore za boju – čunjiće. Korištenje te tri primarne boje za stvaranje boja naziva se *RGB* prostorom boja. Postoje i druge primarne boje koje bi se mogle koristiti, ali pomoću ove tri moguće je pokriti većinu ljudskog spektra boja. [13]

RGB metoda služi samo za raspoznavanje crvenih, plavih i zelenih objekata na slici primljenoj s web-kamere. Prepoznavanje boja vrši se po sljedećem izrazu: [14]

$$f\ R,x,y\ = \begin{matrix} 1 \leftarrow\ f\ G,x,y\ +50 < f\ R,x,y\ \ \wedge\ f\ B,x,y\ +50 < f\ R,x,y \\ 0 \leftarrow\ f\ G,x,y\ +50 > f\ R,x,y\ \ \vee\ f\ B,x,y\ +50 > f\ R,x,y \end{matrix}$$

$$f\ G,x,y\ = \begin{matrix} 1 \leftarrow\ f\ B,x,y\ +50 < f\ G,x,y\ \ \wedge\ f\ R,x,y\ +50 < f\ G,x,y \\ 0 \leftarrow\ f\ B,x,y\ +50 > f\ G,x,y\ \ \vee\ f\ R,x,y\ +50 > f\ G,x,y \end{matrix}$$

$$f\ B,x,y\ = \begin{matrix} 1 \leftarrow\ f\ R,x,y\ +50 < f(B,x,y)\ \wedge\ f\ G,x,y\ +50 < f(B,x,y) \\ 0 \leftarrow\ f\ R,x,y\ +50 > f(B,x,y)\ \vee\ f\ G,x,y\ +50 > f(B,x,y) \end{matrix}$$

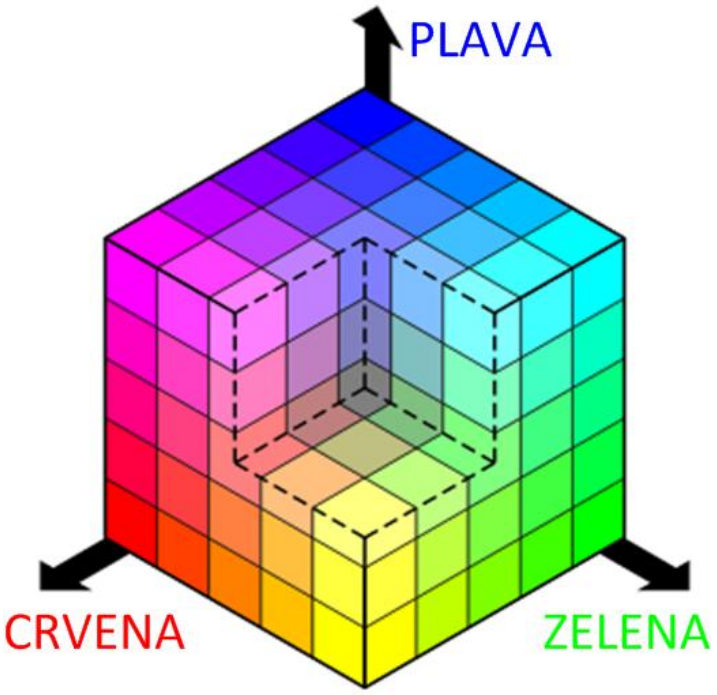
Gdje je : *x,y* – kooordinate piksela na slici

R,G,B – crvena, zelena i plava boja

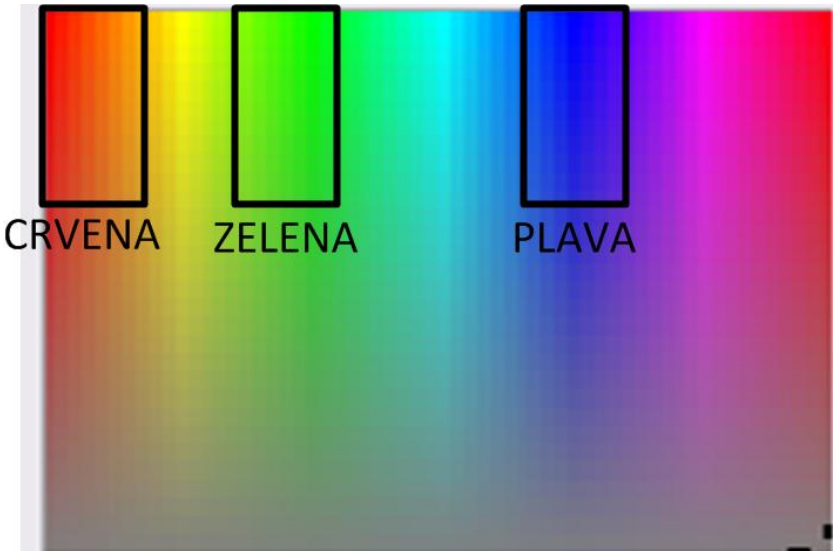
f(R,x,y) – vrijednost crvene boje piksela na koordinatama [x,y]

f(G,x,y) – vrijednost zelene boje piksela na koordinatama [x,y]

f(B,x,y) – vrijednost plave boje piksela na koordinatama [x,y]



Slika 26. RGB prostor boja [13]



Slika 27. Boje koje RGB metoda uspješno raspoznaje

Ova metoda prepoznavanja crvene, zelene i plave boje radi dobro ako je osvjetljenje plohe koju web-kamera snima dobro te ako web-kamera može uhvatiti pravu boju predmeta. Problem ove metode javio se pri promjeni web-kamere gdje se na mjesto prve kamere koja je dobro prepoznavala boje pomoću ove motode stavila druga kamera koja ne uspijeva ispravno uhvatiti boje predmeta, te se zbog toga ova metoda više ne može primijeniti u procesu automatskog raspoznavanja boja.

3.3.1.6.2. HSV metoda raspoznavanja boja

Zbog problema prepoznavanja boja RGB metodom u izradi rada implementiran je algoritam prepoznavanja boja pomoću HSV metode.

Metoda HSV nastala je zbog toga što čovjek teško može pogoditi o kojoj se boji radi ako ima samo vrijednosti crvene, zelene i plave i teško može reći od kolikih udjela primarnih boja se sastoji neka boja. To je ujedno i najveći nedostatak RGB metode pri ljudskoj detekciji boja i detekciji boja pomoću računalnog vida. Metodom HSV imamo poboljšanu reprezentaciju boja zbog toga što je RGB prostor boja u obliku kocke zamijenjen prostorom boja u obliku valjka. U tom valjku tonovi boje kreću od neutralne osi valjka u radijalnom smjeru prema van tako da na rubovima valjka leže potpuno zasićene boje. U valjku boja se u podnožju nalazi crna boja, a na vrhu bijela boja. [15]

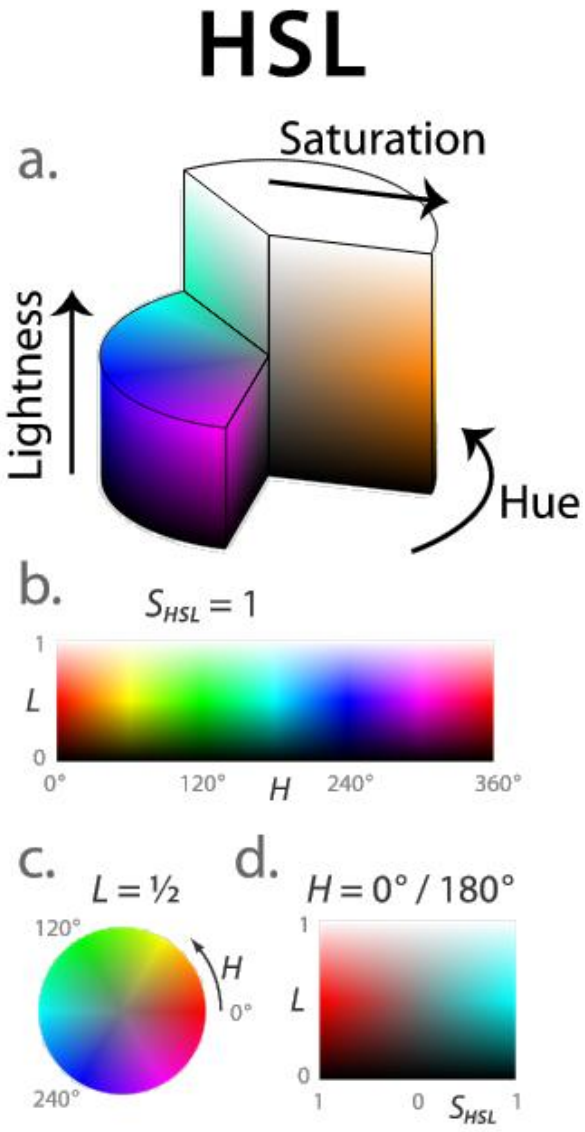
Slika 28. prikazuje prostor boja u HSL formatu te na njoj dio:

a) predstavlja izrezani trodimenzionalni model boja gdje vidimo da boje ovise o tri parametra: zasićenosti bojom (*eng. saturation*), svjetlini boje (*eng. lightness*) i tonu boje (*eng. hue*). Vidimo da se u podnožju valjka kreću tamnije boje s crnom na dnu, dok se prvi vrhu valjka kreću svjetljije boje s bijelom na vrhu;

b) predstavlja dvodimenzionalni prikaz razvijenog oplošja valjka na kojem vidimo promjene boje u ovisnosti o promjeni svjetline i tona. U ovom je prikazu zasićenost bojom konstantna i iznosi jedan;

c) predstavlja dvodimenzionalni prikaz presjeka valjka na kojem vidimo promjene boje u ovisnosti o promjeni zasićenosti i tona boje. U ovom je prikazu svjetlina boje konstantna i iznosi jednu polovinu;

d) predstavlja dvodimenzionalni prikaz na kojem vidimo promjenu boje u ovisnosti o promjeni svjetline boja i zasićenosti boje. U ovom prikazu ton boje je konstantan te iznosi 0 i 180 stupnjeva. Na slici vidimo crvenu boju i njenu komplementarnu boju svijetlomodru. [15]

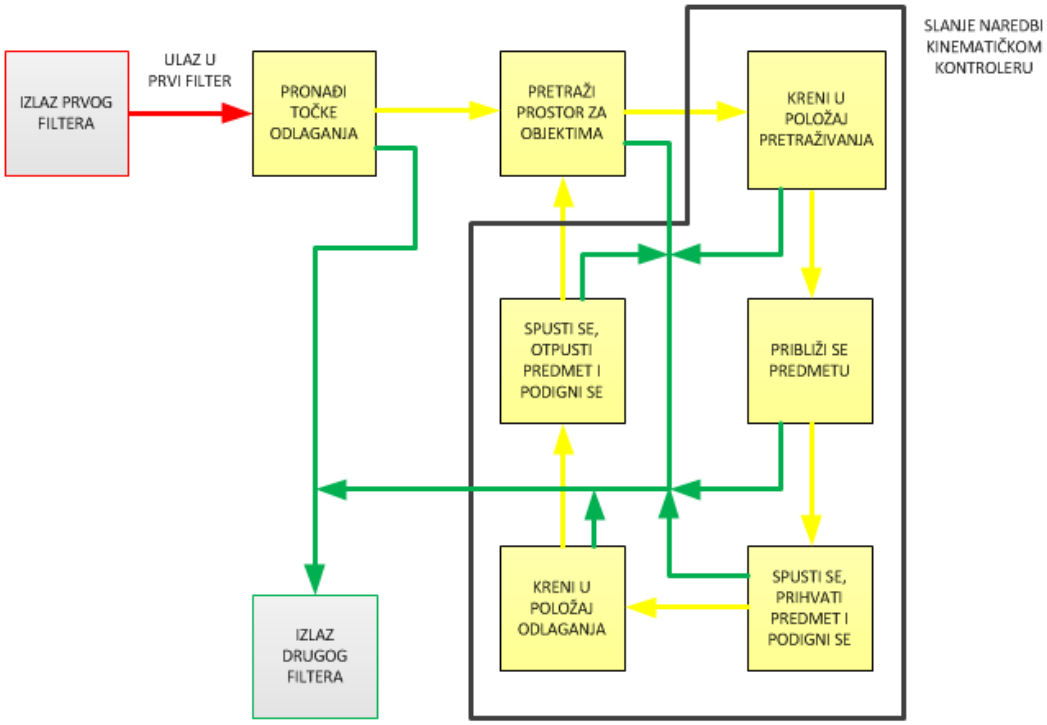


Slika 28. Prikaz prostora boja pomoću HSL metode [15]

Korištenjem ove metode za odabir boje koju želimo da kamera prepoznaje te njeno kasnije pronalaženje uvelike se poboljšala detekcija boja. Kako bi boja bila uspješno prepoznata, svjetlina boje mora biti unutar raspona od 40% do 60%, te zasićenost bojom mora bit veća od 60%.

3.3.2. Struktura drugog filtera

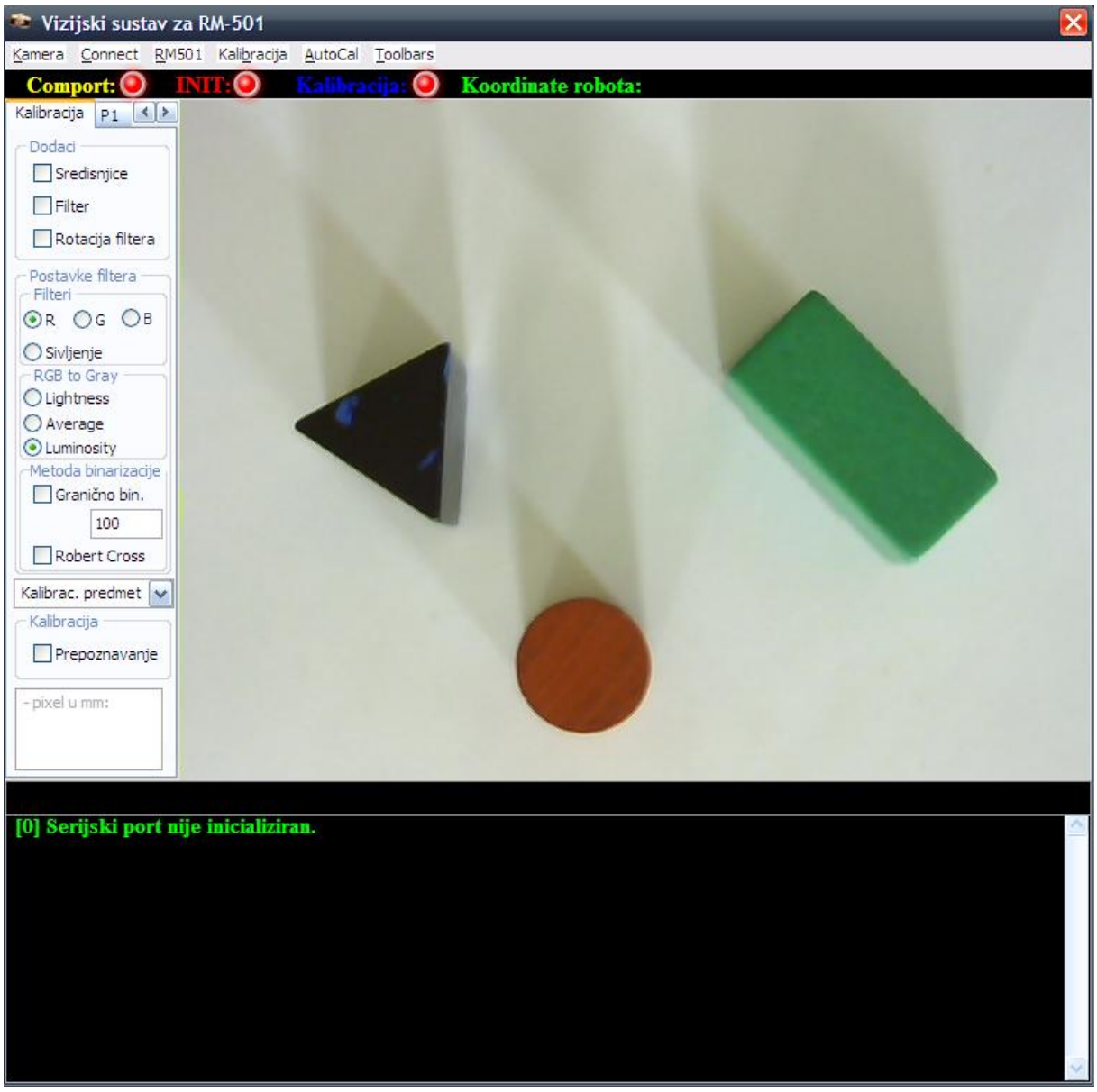
Ulazni parametar strukture drugog filtera je slika koju daje izlaz strukture prvog filtera. Struktura drugog filtera koristi se samo prilikom automatskog sortiranja objekata. Kada automatsko sortiranje objekata nije aktivno tada ni struktura drugog filtera nije aktivna, tj. slika koju struktura dobije na ulazu direktno se prosljeđuje na izlaz strukture. Kada je pokrenuto automatsko sortiranje, tada se u ovoj strukturi odvijaju operacije traženja središta za odlaganje prihvaćenih predmeta, slanje naredbi robotu za kretanjem na koordinate pretraživanja ili odlaganja, spuštanja, dizanja i pomicanja prema predmetu te njegova prihvata.



Slika 29. Struktura drugog filtera

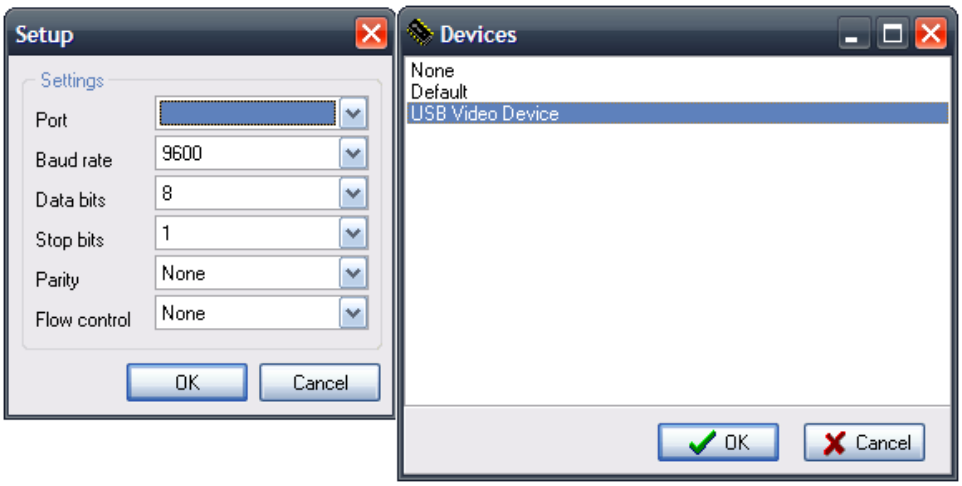
4. UPUTE ZA PRIMJENU PROGRAMA

Prilikom pokretanja programa pojavit će se sučelje programa gdje će nam na vrhu programa biti prikazane statusne diode, dok će na dnu biti prikazana konzola i statusni dio. S lijeve strane prozor s dostupne tri kartice – kalibracija, P1 (Program 1) i P2 (Program 2), a s desne strane bit će prikazana slika dobivena s web-kamere



Slika 30. Izgled programskog sučelja za robot Mitsubishi RM501

Zbog toga što se USB priključak koji se spaja na kinematički kontroler može spojiti na bilo koji USB *port* računala, potrebno je ručno odabrati na kojem je *portu* priključak spojen. Isto tako, ako se ne pojavi slika na desnom dijelu programskog sučelja, potrebno je ručno odabrati koju kameru želimo koristiti kao video ulaz u program. Pokretanje ručnog odabira *porta* vrši se pritiskom na gumb *Connect* koji se nalazi na vrhu programskog sučelja. Pokretanje ručnog odabira kamere vrši se pritiskom na gumb *Kamera* koji se nalazi na vrhu programskog sučelja.



Slika 31. Izgled prozora za odabir porta i kamere

Nakon odabira *porta* pojavljuje se status o tome na koji *port* smo trenutno spojeni te se statusna dioda na vrhu prozora mijenja u zelenu boju. Ako smo odabrali dobar *port* preko kojeg smo spojeni na kinematički kontroler, tada ćemo dobiti povratnu informaciju o tome je li robot inicijaliziran ili nije. Ako je robot inicijaliziran, tada će se boja statusne diode na vrhu prozora promijeniti u zelenu te će se na vrhu prozora ispisati trenutne koordinate robota. Ako robot nije inicijaliziran, tada inicijalizaciju robota možemo pokrenuti tako da u konzolu upišemo naredbu *init* ili da na vrhu prozora odaberemo *RM501* padajući izbornik i odaberemo *init*.

Nakon što smo spojeni s kinematičkim kontrolerom, od njega možemo zatražiti različite informacije pomoću naredbi koje su navedene u Tablica 1.

Tablica 1. Popis naredbi konzole programa

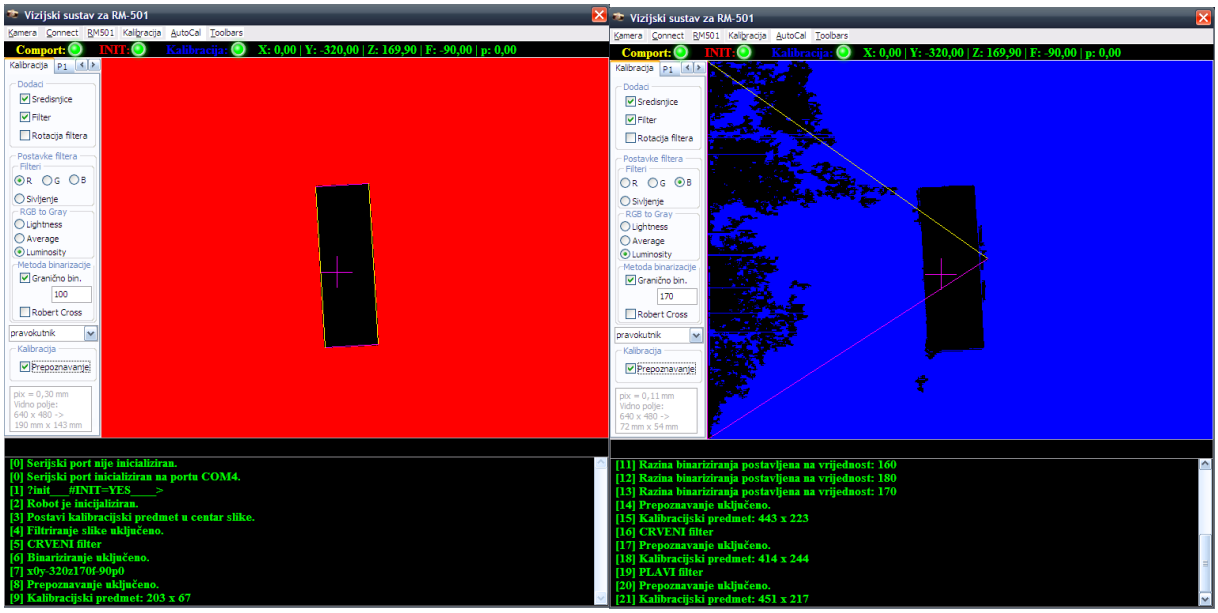
***x**100**y**50**z**80**f**-80**p**0*	zadavanje apsolutnih koordinata
***y**100*	
***dx**100**dy**50**dz**80**df**-80**dp**0*	zadavanje relativnih koordinata
***dy**20*	
SPEED=4	promjena brzine kretanja robota
OPEN	otvaranje hvataljke
CLOSE	zatvaranje hvataljke
INIT	pokretanje inicijalizacije robota
HOME	slanje robota u HOME položaj
?R	vraća vrijednosti vektora R
?Q	vraća vrijednosti vektora Q
?A	vraća vrijednosti vektora A
?INIT	ako je robot inicijaliziran vraća YES, u suprotnom NO
?SPEED	vraća trenutnu brzinu robota

Tablica 2. Popis mogućih pogrešaka koje šalje kinematički kontroler

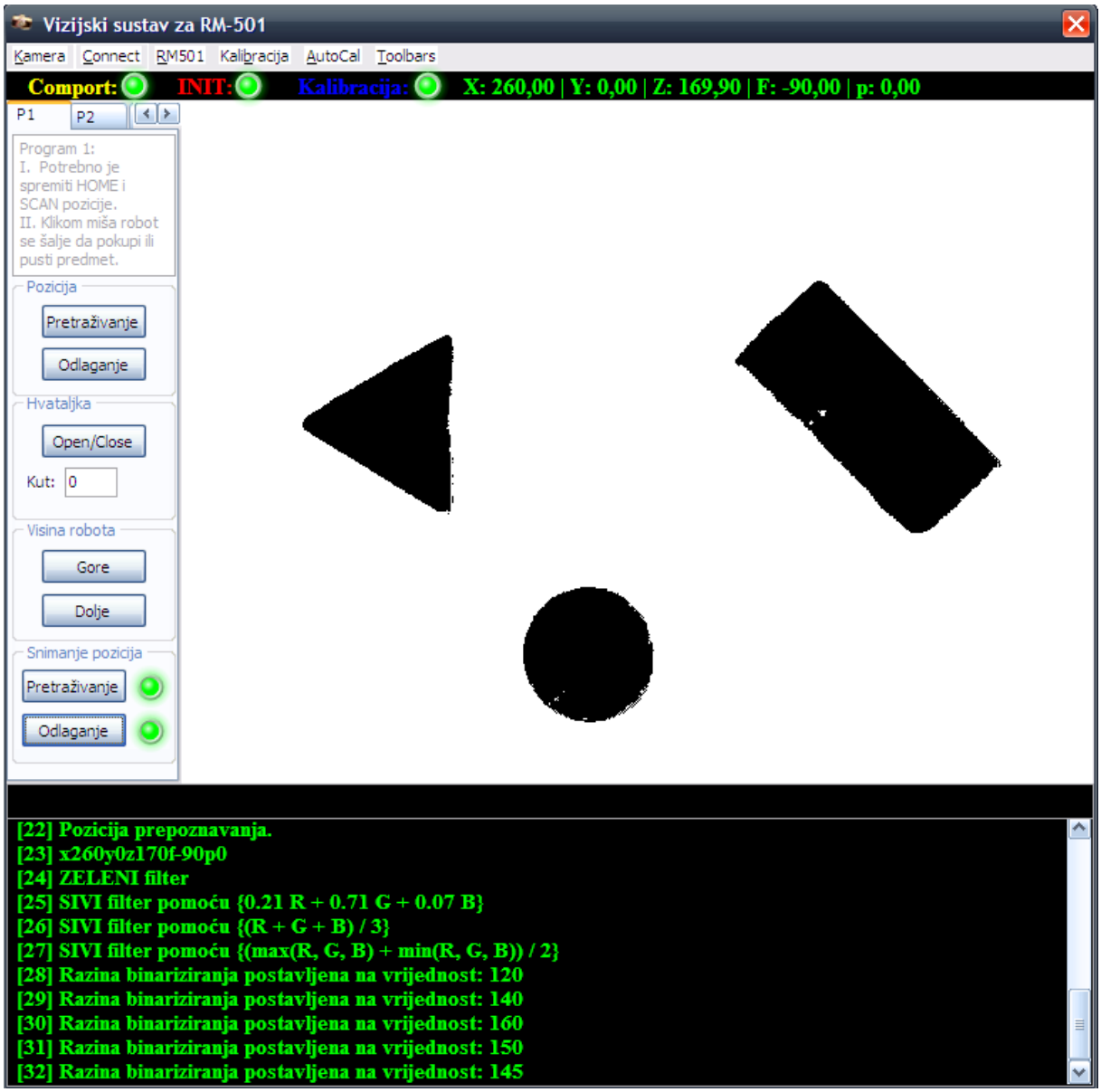
E0	Nema pogreške
E1	Parametar a1 je izvan dopuštenog raspona
E2	Parametar a2 je izvan dopuštenog raspona
E3	Parametar a3 je izvan dopuštenog raspona
E4	Parametar a4, a5 su izvan dopuštenog raspona
E5	Ulazni parametar je izvan dopuštenog raspona
E6	Točka je izvan dopuštenog raspona
E7	Robot nije inicijaliziran
E8	Opasnost od sudara
E9	Nepoznata naredba

Nakon spajanja s robotom te po potrebi njegove inicijalizacije potrebno je pokrenuti kalibraciju kamere. Kalibracija kamere može se izvršiti dovođenjem robota do mjesta gdje se nalazi kalibracijski crtež pomoću naredbi u konzoli, ili se može pokrenuti automatska kalibracija gdje robot sam odlazi na predodređeno mjesto, uključuje filter i binarizaciju te odabire kalibracijski objekt. Korisnik samo mora pokrenuti prepoznavanje na dnu kalibracijske kartice kada vidi da se kamera dovoljno umirila. Automatska kalibracija pokreće se odabirom *AutoCal* gumba koji se nalazi na vrhu programskog prozora. Ako korisnik sam odluči vršiti kalibraciju kamere, tada mora - nakon što je doveo kameru robota do kalibracijskog objekta - označiti *Filter* kvadratić za izbor. Automatski odabrani filter bit će crveni, ali ako korisnik programa želi promijeniti vrstu filtera, tada mora odabrati jednu od ponuđenih opcija u postavkama filtera. Nadalje, mora odabrati binarizaciju te vrstu korištenog kalibracijskog objekta. Ako korisnik mora ponovno učitati bazu objekata, tada mora odabrati gumb *Kalibracija -> Učitaj predmete* na vrhu programskog prozora. Korisnik isto tako može

ispisati trenutno spremljene predmete odabirom *Kalibracija -> Ispiši predmete*. Nakon odabranog kalibracijskog predmeta korisnik može označiti kvadratić za izbor *Prepoznavanje* čime će pokrenuti kalibraciju. Rezultat kalibracije bit će ispisan na dnu kartice *Kalibracija* gdje će pisati informacije o tome koliko jedan piksel na slici iznosi mm u stvarnosti te koliki prostor trenutno kamera vidi. Ako je kalibracija loša, tada je potrebno promijeniti granicu filtracije ili promijeniti vrstu korištenog filtera. S trenutnom kamerom jedan piksel iznosi približno 0,3 mm na visini robota od 170mm. Nakon izvršene kalibracije kamere korisnik može pokrenuti Program 1 ili Program 2.



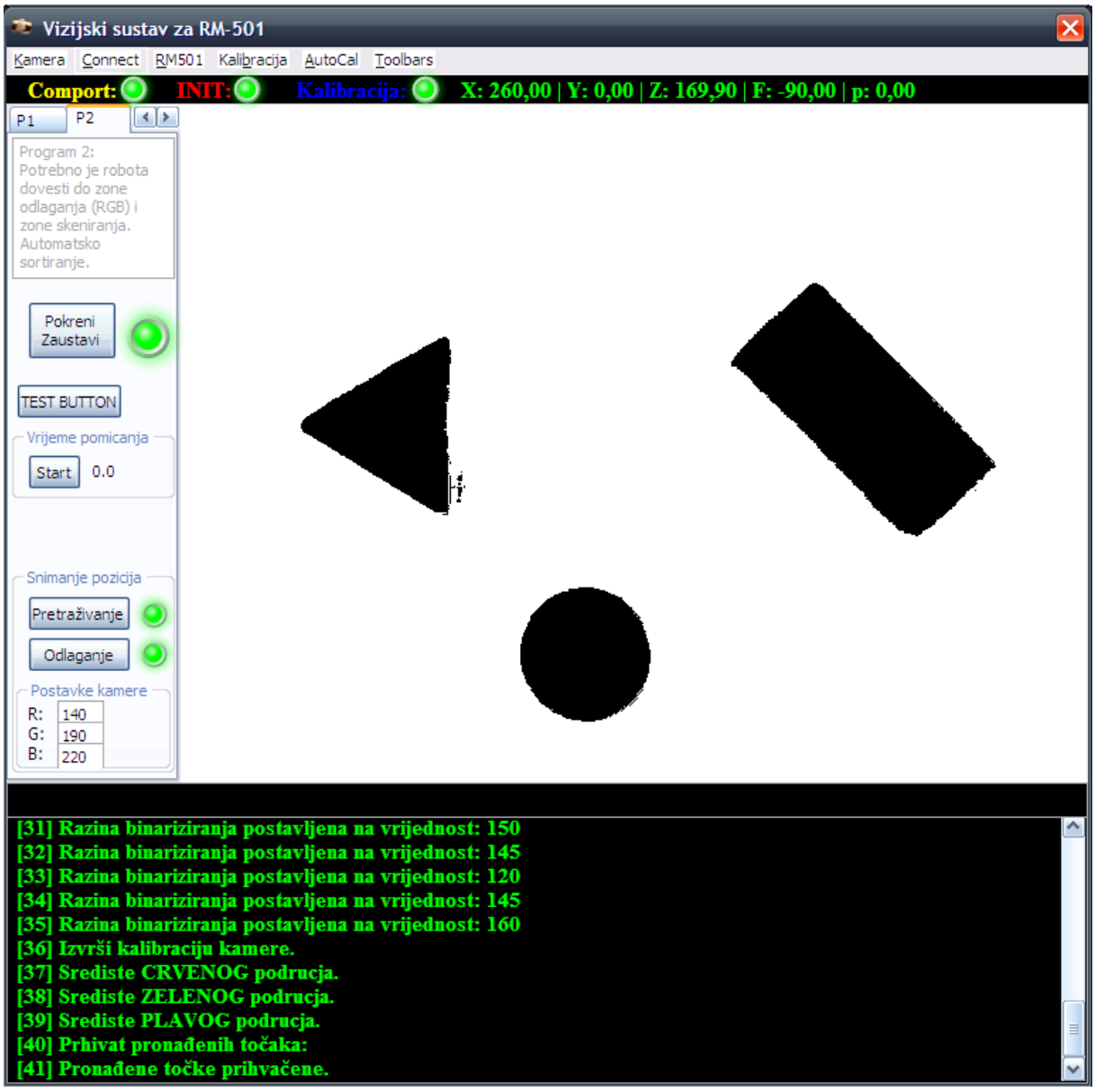
Slika 32. Ispravna i neispravna kalibracija kamere



Slika 33. Izgled Programa 1

Kada korisnik odabere karticu Program 1, prvo mora pohraniti vrijednosti koordinata pretraživanja i odlaganja. Nakon što pohrani te koordinate, tada će se korisniku prikazati Program 1. U Programu 1 korisnik može klikom miša na slici kamere slati robotu naredbe kamo da se kreće. Pomoću gumba koji se nalaze u sekciji pozicija robota može slati na spremljene koordinate pretraživanja i odlaganja. Pomoću gumba *Open/Close* koji se nalazi u sekciji *Hvataljka* korisnik može otvoriti ili zatvoriti hvataljku na robotu. Nadalje, u toj sekciji korisnik može podesiti kut zakreta hvataljke robota. U zadnjoj sekciji *Visina robota* korisnik

može odabrati želi li robota poslati dolje u središte kamere da pokupi neki objekt ili ga želi podići.

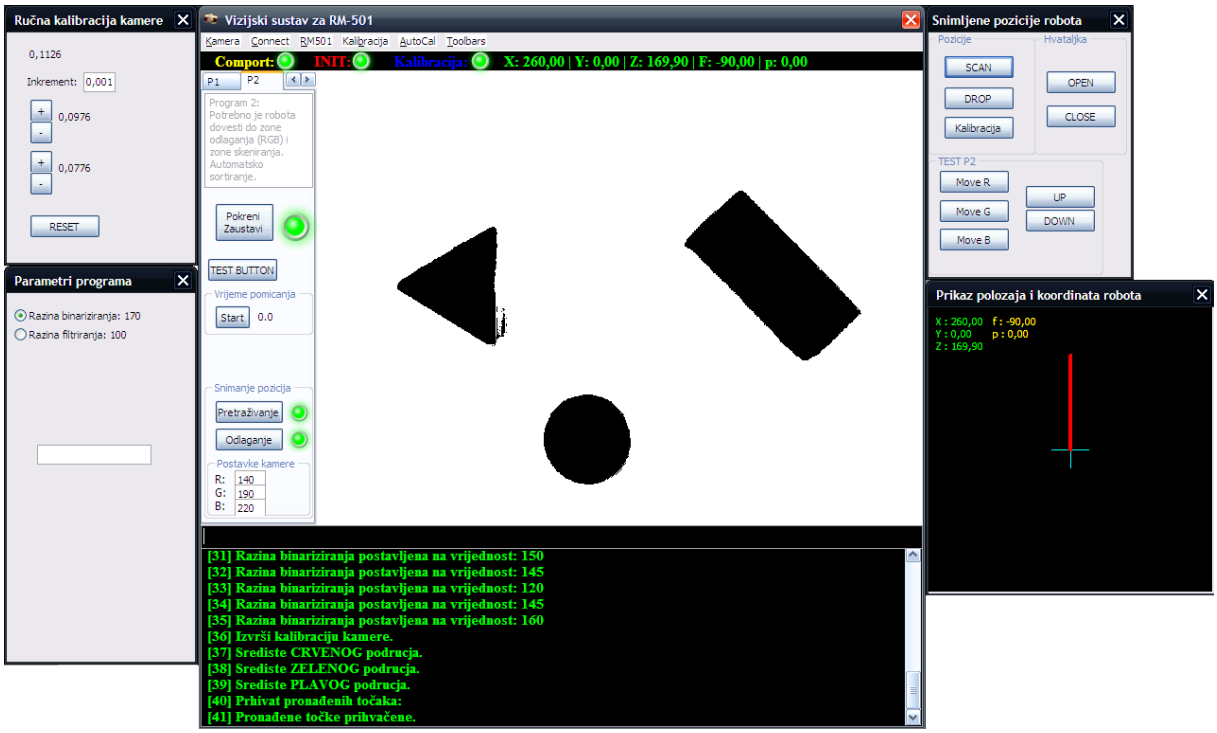


Slika 34. Izgled Programa 2

Kada korisnik odabere karticu Program 2, prvo mora pohraniti vrijednosti koordinata pretraživanja i odlaganja. Nakon što pohrani te koordinate, tada će se korisniku prikazati Program 2. Korisnik također mora ručno podesiti granične vrijednosti binarizacije za crveni, zeleni i plavi filter koje se nalaze na dnu te kartice u sekciji postavke kamere. Nakon što je korisnik podesio granice i spremio pozicije, pokreće program klikom na gumb *pokretanje/zaustavljanje*. Nakon pokretanja programa on će izvršiti pregledavanje pozicije

odlaganja za točkama na kojima će odlagati predmete te će pronađene točke prikazati korisniku. Ako točke nisu točno prepoznate, što je moguće zbog loše kvalitete kamere, tada je potrebno na sekciji prihvat središta odabrati gumb *NE,* u suprotnom ako su sekcije ispravno odabrane, tada je potrebno odabrati gumb *DA.* Ako je korisnik odabrao *NE,* tada program očekuje od korisnika da ručno - klikom miša - odabere tri središta odlaganja na slici kamere. Korisnik može ovaj odabir ponavljati koliko puta želi dok nije zadovoljan pritiskom na gumb *NE.* Nakon što je zadovoljavajuće odabrao središta odlaganja, korisnik mora pokrenuti *timer* za mjerenje vremena kretanja robota od točke prepoznavanja do točke odlaganja. Ručno mjerenje je potrebno zbog toga što robot ne raspoznaje kreće li se od početne do zadane točke ili se već nalazi u zadanoj točki. Od korisnika se traži da pritisne *START* za početak *timera* i *STOP* za kraj kada se robot prestane gibati. Ako korisnik misli da je vrijeme prijenosa bilo ispravno, tada treba na prozoru koji će se stvoriti odabrati *OK,* u suprotnom potrebno je odabrati *Cancel* i ponoviti mjerenje vremena prijenosa. Nakon što korisnik odabere *OK,* pokrenut će se program automatskog sortiranja gdje će robot uzimati predmete s pozicije preuzimanja i odnijeti ih na poziciju odlaganja. Nakon izvršenja automatskog prepoznavanja program će se sam zaustaviti i ispisati koliko je predmeta prenio.

Dodatne funkcije programa korisnik može aktivirati otvaranjem padajućeg izbornika na vrhu *Toolbars*. U njemu korisnik može pokrenuti dodatne prozore kao što su prikaz robota, pozicije, kalibracija kamere, parametri programa ili može otvoriti sve navede prozore pritiskom na gumb *Aktiviraj SVE*. U tome padajućem izborniku korisnik može odabrati *Save Display* pomoću kojeg će se trenutni *display* sinimiti u mapu u kojoj se nalazi izvršna datoteka programa pod imenom koje će se sastojati od datuma i vremena kada je gumb pritisnut.



Slika 35. Izgled programa s pokrenutim dodatnim funkcijama

Ručna kalibracija kamere koristi nam za ručno podešavanje odstupanja kamere po smjeru x i y. Ovo je najbolje podešavati uz pogrenuti Program 1 jer se tamo vidi u kojem smjeru zadana kretnja robota odstupa od tražene. U sekciji *Parametri programa* moguće je promijeniti neke parametre programa. U položaju i koordinatama robota vidimo trenutni položaj robota i njegove koordinate te vizualni prikaz njegova kuta zakreta. U snimljenim pozicijama robota postoje unaprijed spremljene pozicije pregledavanja i odlaganja, moguće je otvarati i zatvarati hvataljku, spuštati i podizati robot te testirati koordinate odlaganja koje su spremljene u Programu 2.

5. ZAKLJUČAK

Vizijski sustavi sve se više koriste u industriji gdje zamjenjuju različite kontaktne i bezkontaktne senzore te ljudski rad. Pomoću njih procesi proizvodnje i montaže mogu se uvelike pojednostaviti i ubrzati. Problem vizijskih sustava jest taj da je vizijski sustav toliko dobar koliko je dobra kamera koja se koristi za akviziciju slike.

U ovom radu programiran je i eksperimentalno provjeren vizijski sustav robota Mitsubishi RM501. Kao kamera vizijskog sustava korištena je web-kamera koja se u određenim uvjetima pokazalo lošom opremom tj. odabirom. Neki od nedostataka korištene web-kamere su: uvelike je ovisna o osvjetljenju – bolje radi tijekom dana kada je prisutno prirodno svjetlo nego navečer i noću; nema mogućnosti samopodešenja fokusa – moguće je pretraživati područje samo s jedne visine s malim odstupanjima, što je loše jer je nemoguće obaviti približavanje i detaljniju inspekciju pronađenog objekta; rezultat njene unutarnje funkcije kojom prepoznaje boje mogu biti iskrivljene vrijednosti boja, čime se onemogućava prepoznavanje boja objekta. Unatoč ovim nedostacima kamere, programirani vizijski sustav može vršiti kalibraciju kamere, upravljanje robotom klikom miša na prikaz slike s kamere te automatsko uzimanje predmeta s jednog položaja i njihovo prenošenje na drugi položaj.

Prednosti ovog vizijskog sustava su da je on programiran kao modularna aplikacija, čime se olakšava buduće mijenjanje dijelova programa, te to da nije vezan uz točke okoline – kalibracija, preuzimanje i odlaganje mogu se vršiti na bilo kojem mjestu unutar radnog prostora robota. Neki od trenutnih nedostataka ovog vizijskog sustava su ti što je za sada nemoguće ispravno prepoznavanje boje predmeta te to što sustav automatskog sortiranja radi u vremenski podoptimalnom području. Problem je u tome što robot ne daje povratnu informaciju o svom gibanju, a testiranje i izrada pregledne tablice nisu bili predviđeni ovim radom, tako da određivanje vremena kretanja iz jedne pozicije u drugu ostaje na korisniku. Ako korisnik procijeni vrijeme kretanja iz jedne pozicije u drugu prekratkim, može doći do zaustavljanja programa ili preskakanja određenih operacija, a ako ga procijeni predugim, tada vrijeme izvođenja može biti veoma dugo.

Buduća poboljšanja sustava trebala bi uključivati ugradnju kvalitetnije kamere - čime bi se ubrzao i olakšao proces prepoznavanja zbog toga što ne bi postojala potreba za kompleksnim algoritmima filtracije slike koji su trenutno implementirani; testiranje sustava sa

stacionarnom kamerom - čime bi se riješio trenutni problem vremena gibanja robota iz jedne pozicije u drugu; poboljšanje sustava kalibracije kamere - čime bi se riješio problem izobličenja slike zbog sfernih aberacija kamere i njena neokomita položaja.

Sve u svemu, ovaj rad može poslužiti kao osnova za daljnje razvijanje vizijskih sustava koji će se baviti prepoznavanjem i sortiranjem predmeta, s mogućnošću kasnije primjene u industrijskom okruženju.

PRILOZI

I. CD-R disc

LITERATURA

[1] 3D-skeneri koje koristi tvrtka Topomatika. Web-stranica posjećena 20. prosinca 2012. na web-adresi: http://www.topomatika.hr/atos.html

[2] Popis komercijalnih primjena vizijskih sustava. Web stranica posjećena 21. prosinca 2012. na web-adresi: http://www.cs.ubc.ca/~lowe/vision.html

[3] Jerbić B., predavanja iz kolegija „Vizijski sustavi“, Zagreb, 2012.

[4] Slika kompjuterskog vizijskog sustava preuzeta 25. lipnja 2012. s web-adrese: http://www.high- techdigital.com/integration/Images_Integration/System_Integration_E.gif

[5] Malik. J., online-tečaj: “Computer vision“, 2012.

[6] Slika preuzeta 15. siječnja 2013. s web-adrese: http://en.wikipedia.org/wiki/Lenna

[7] Crneković M., predavanja iz kolegija „Industrijski i mobilni roboti“, Zagreb 2012.

[8] Hranj N., završni rad: „*Pomoćni kontroler robota Mitsubishi RM501*“, Zagreb 2011.

[9] Informacije o web-kameri preuzete su 25. lipnja 2012. s web-adrese: http://www.canyon-tech.com/products/voip/webcams/CNR-WCAM513#pr-switcher

[10] Video lab tvrtke Mitov preuzet je 10. siječnja 2012. s web-adrese: http://www.mitov.com/html/videolab.html

[11] ComPort Library koji je napisao Dejan Crnila preuzet je 25. lipnja 2012. s web-adrese: http://sourceforge.net/projects/comport/files/comport/4.10/

[12] Algoritmi za pretvaranje slike u boji u sive slike preuzeti su 2. siječnja 2013. s web-adrese: http://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/

[13] Informacije o RGB formatu boja i ljudskom vidnom spektru preuzete su 5. siječnja 2013. s web-stranice: http://en.wikipedia.org/wiki/Color_model

[14] Informacije o RGB metodi prepoznavanja boja preuzete su 5. siječnja 2013. s web-stranice: http://myopencv.wordpress.com/2009/06/13/detecting-colors-using-rgb-color-space/

[15] Informacije o HSL formatu boja preuzete su 20. siječnja 2013. s web-stanice: http://en.wikipedia.org/wiki/HSL_and_HSV

[16] Informacije o implementaciji HSL metode u programski jezik Delphi preuzete su su 20. siječnja 2013. s web-stranice: http://www.efg2.com/Lab/Graphics/Colors/HSV.htm

[17] Informacije o implementaciji boja i dll datoteka u programski jezik delphi preuzete su 6. siječnja 2013. s web-strance: http://delphiforfun.org/

[18] Informacije o brzom pristupanju boja u programskom jeziku Delphi preuzete su 20. prosinca 2012. s web-stranice: http://www.delphibasics.co.uk/